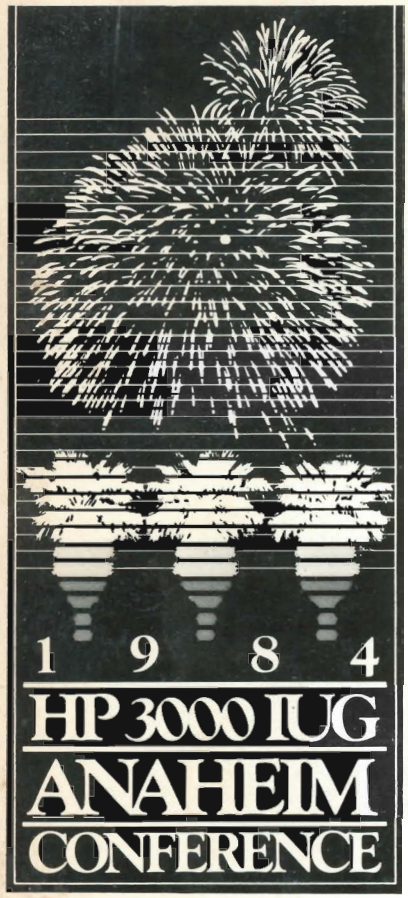


HOSTED BY THE
SOUTHERN CALIFORNIA
REGIONAL USERS GROUP

HP 3000 INTERNATIONAL USERS GROUP

1984 HP 3000 INTERNATIONAL USERS GROUP
DISNEYLAND HOTEL,
FEBRUARY 26–MARCH 2
ANAHEIM, CALIFORNIA, USA

PROCEEDINGS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

EVAN RUDDEROW

CONFERENCE PROCEEDINGS

of the

HP3000 INTERNATIONAL

USERS GROUP

February 27 - March 2 1984
Anaheim, California
USA

Program committee
--
Frank Vogt
Ann E. Paul
Charles Finley

Prepared by the Southern California Regional User's Group

SCRUG HOST COMMITTEE MEMBERS

Charles Finley Jr.
Judith S. Fisher
Carm Gravagno
Larry Haftl
Arline King
Frank Irvine
Jane Knight
Tom Knight
Doug Mecham
Anne E. Paul
James Robinson
Dr. Halmuth H. Schaefer
Charles Shimada
Bruce Toback
Vicki Toback
Frank Vogt
Oren "Ken" Williams
James Wylie

IUG Staff

William Crow, Association Manager
Aldena Duval, Membership Services Manager
Renaye Lee, Conference Manager
Ben Lloyd, Technical Manager
Jeff Miotke, International Liaison Manager
Nancy Trankle, Membership Activities Manager
Linda Henson, Financial and Administration Manager
Betty Martin, Publications Manager

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

LIST of PRESENTATIONS – by SPEAKER

BUSINESS (BU)

DATA BASE (DB)

DATA COMMUNICATIONS (DC)

GRAPHICS (GR)

LANGUAGE SUPPORT (LS)

OFFICE AUTOMATION (OA)

OPERATIONS (OP)

PERSONAL COMPUTERS (PC)

QUALITY ASSURANCE (QA)

SYSTEM MANAGEMENT (SM)

USER APPLICATIONS (UA)
(System Development)

MISCELLANEOUS (MI)

PERSONAL COMPUTER NETWORKING

Paul T. Antony
Hewlett-Packard

Although Local Area Networking technology is still at an early stage, it is already enjoying rapid growth. Computing environments increasingly require the interworking of many different data processing devices, each with its own degree of intelligence, each with its own advantages in terms of functionality and cost, each capable of operating more or less independently of any other system component, and, increasingly often, each supplied from a different source in order to take advantage of the rapid developments in information processing technology. Providing such an environment through integration within a local area network will offer the users of mini computer based systems new power and versatility, with which they will be able to offer a real price/performance challenge to traditional mainframe based systems. An integrated system based on a LAN also offers the hope of a smoother path for upgrading overall systems capability through replacement of individual functional units as newer and more powerful facilities become available.

Few areas in the data communications world have seen as much recent technological innovation and new commercial offerings, as the area of Local Area Networking.

As the cost of individual processing components falls, organizations are aquiring greater numbers of separate "specialized" computer-based systems. Each system focused on meeting specific user needs for higher worker productivity, better computer accessibility, and faster responsiveness. The benefits of conveniently interconnecting these systems to allow them to share common resources is becoming increasingly important. This includes both software resources (such as databases, development tools, reports, and office style memos), and hardware resources (such as expensive high speed printers and large plotters.)

A local area network is a data communications system that can be used to provide the level of interconnection described above -- ideally providing these services transparent to the end users of the different systems being connected.

Some general characteristics of this type of network are:

- o HIGH DATA RATES - One megabit/sec or higher. Usually, at least 10 megabits/sec. To allow fast, multiple station access with transparent network operation.
- o LOW TRANSMISSION ERROR RATES - Networks must realiably accomodate heavy data transmission traffic. Should an error occur, a network station should detect it and institute a recovery.
- o LIMITED GEOGRAPHICAL COVERAGE - Distances can vary from a few hundred feet to several miles. Generally, LAN's span less than two kilometers.
- o RELIABLITY AND HIGH AVAILABILITY - Networks are highly user interactive- LAN should have sufficient capacity to handle bursty traffic without long system delays. LAN should remain unaffected by individual failures -- and network maintenance should be done with minimal interuption of network services.
- o FLEXIBLE TOPOLOGY - Network should be flexible so that as your organization changes you can readily attach, disconnect, and delete stations without operational changes and disruption of network service.

- SINGLE ORGANIZATION OWNERSHIP - LAN's are usually wholly owned by a single organization, with gateways to allow communications to other organizations.

There are a broad spectrum of LAN products available, and finding the one that best suits your particular needs can be difficult. Anticipated use will determine which network type best matches your application requirements. For example, some applications such as office automation, entail text editing and file processing, which can occur at adequate rates over short (room) connection distances using low cost desktop computers. On the other hand, general purpose data and message communication applications, with or without office automation, demand more expensive minicomputers interconnected over relatively long (floor) distances. Finally, for very high speed, complex engineering and scientific computations, you might need costly superminicomputers interconnected to nodes scattered throughout one or more buildings.

To select the Local Area Network that best fits your needs, you need to have a basic understanding of these LAN criteria.

1) NETWORK TOPOLOGIES

2) NETWORK ACCESS METHODS

3) MEDIA ALTERNATIVES

LOCAL AREA NETWORK TOPOLOGIES

A) STAR TOPOLOGY

A star network consists of a central node to which each of the host system devices are connected (Fig 1). The central node acts as a routing switch for data arriving at the central node from each of the host connections. A star network simplifies access control and routing decisions required within attached hosts. However, throughput performance and reliablility of the entire network relies on the operation of the single central node.

Primary Advantage(s):
- SIMPLE/LOW COST NODES

Primary Disadvantage(s):
- SINGLE POINT OF FAILURE
- PERFORMANCE DEPENDENDENT ON CAPACITY OF CENTRAL NODE

Primary Application:
- CENTRALIZED PBX TYPE SYSTEMS -- WORKSTATION TO SYSTEM COMMUNICATIONS

B) RING TOPOLOGY

A ring topology seeks to eliminate the dependency on a central, controlling node of the star network without sacrificing the relative simplicity of the other nodes (Fig 2). In a ring network a single communications path is shared amongst, all attached nodes. This path is unidirectional and provides for the transfer of discrete packets of data. Each packet of data is injected into the ring from one node to the next in a predefined direction. There are no routing decisions to be made in this topology. The sending node simply transmits its message towards its next neighbor node in the ring and the message then passes around the ring until it reaches the node for which it was intended. Each node acts as a regenerative repeater of receiving packets, generally introducing one or more bits of delay as it does so. The only routing requirement made of each node is that it be able to recognize those messages that are intended for it, by examination of the node address contained in each data packet. Then, dependent on the control strategy and implementation, the receiving node may remove the packet from the ring or pass the packet on back to the transmitting node with an acknowledgement field marked to indicate whether or not the packet was accepted. In this case, the original transmitting node removes the packet from the ring.

The inherent weakness in this structure is that the transmission path relies not only on the integrity of the transmission medium, but also on that of the ring interface which is an active component of the network. A failure in either of these two areas could paralyze the entire network. Techniques have been devised which can detect repeater failures and switch those units out, while allowing the remainder of the network to function normally -- however these devices increase both the cost and the complexity of each node.

Primary Advantage(s):
- ELIMINATE DEPENDENCY ON CENTRAL NODE
- NODES REMAIN RELATIVELY SIMPLE

Primary Disadvantage(s):
- SUSCEPTIBLE TO NETWORK FAILURES DUE TO SINGLE NODE FAILURE
- BUILDING PREWIRING CONSTRAINTS DUE TO CIRCULAR GEOMETRIES

Primary Application:
- HIGHSPEED SYSTEM TO SYSTEM COMMUNICATIONS

D) BUS TOPOLOGY

The bus or broadcast network structure is conceptully simpler than that of the ring. The basic structure is linear and derives essentially from traditional computer architecture of input/output channels (Fig 3). The bus medium itself is passive and allows bidirectional transfer of messages. Like the ring, the bus structure does not require any of the attached nodes to make routing decisions. A message simply flows away from the originating node in both directions towards the ends of the bus. The intended destination node must be able to recognize messages that are intended for it and then read the message as it passes by. Each node is attached to the bus medium in a 'T' fashion so that the message signal continues to propagate down the bus whatever the action in or by the attached nodes; there is therefore no requirement for a bus node to absorb and regenerate the message, and no modification or delay is imposed on the information in transit. It is this intrinsic feature of the bus topology which offers the great attraction of simplicity and 'fail safe' operation.

Primary Advantage(s):
- DECENTRALIZED CONTROL
- BUS MEDIUM IS PASSIVE (NO ACTIVE REPEATERS)
- EASY TO PREWIRE BUILDINGS; VERY FLEXIBLE GEOMETRY

Primary Disadvantages(s):
- SIGNAL DEGENERATES OVER DISTANCE (UNLESS REPEATERS ARE INSTALLED)

Primary Application:
- `BURSTY TRAFFIC' SYSTEM TO SYSTEM COMMUNICATION

NETWORK ACCESS METHODS

Currently, there are two dominant but widely varying network access control schemes that have proven successful: carrier sense multiple access with collision detection (CSMA/CD) and token passing.

A) TOKEN PASSING

Token Passing is used predominantly in ring networks. By design, only the node that holds the token at any time can transmit on the network; therefore no access control conflicts can arise. In a ring network, a message token passes from node to node in one direction during the idle network periods. Under strict timing rules, any node wanting network access must acquire the token within a defined interval by altering one of the token's bits on the fly. This node then transmits its message. The sequence for aquiring control of the network using token passing is as follows:

1) node wanting to transmit, waits for token
2) node removes token from network
3) node transmits "information" packet onto network
4) packet travels around network in predefined path
5) receiving node(s) read packet and may set an acknowledgement field on packet trailer
6) packet travels back to transmitting node
7) transmitting node removes packet
8) transmitting node reinserts token onto network

Token passing is highly deterministic and predictable. One can calculate the maximum delay that a station will encounter in gaining network access (this is not possible with CSMA methods, whose channel access times fluctuate randomly).

Another token passing advantage stems from the high transmission efficiency achieved for varied message/packet sizes and data rates. Other key advantages include guaranteed maximum access times to accomodate real time applications, reliable operation under all load conditions and media independence without collision detection mechanisms. However, tokens travel only in one direction in a ring network; if a node misses a token, it must wait until the token makes a complete ring revolution. Additionally, a break in the enclosed ring opens the circuit and destroys the token. The asynchronous operation of token passing can allow message tokens to get lost, destroyed, or degraded in a distributed control topology. Furthermore, strict timing requirements translate into design complexity.

B) COLLISION SENSING MULTIPLE ACCESS/COLLISION DETECTION (CSMA/CD)

Just as token passing dominates in control accessing ring networks, the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) method governs virtually all bus networks. It involves two major operational rules. First when the bus or data channel is busy -- attending to a nodes's message transmission needs -- all other nodes must wait until the channel clears before trying to send their own messages. Second, if multiple nodes then try to transmit simultaneously resulting in message interference or collision, these nodes must stop transmitting and wait for varying delay times before transmitting again. This access method's chief advantages lie in 1) its simplicity - reflected in lower costs per node because the scheme needs no complex or expensive priority access circuits, and 2) its variable length message handling efficiency - more than 99 percent of all messages get through in bursty or intermittent applications (such as office automation). Conversely, for heavy traffic applications such as busy data or reservation processing systems, CSMA/CD incurs higher message collision levels, longer access delays and reduced thoughput. Moreover, it doesn't suit real time needs because no priority mechanism exists: Messages with different degrees of importance compete equally for bus access. In the ultimate worst lockout condition, a node's message continually collides with those of competing nodes and never gains bus access. Several bus networks do not use the collision detection aspect of CSMA/CD because their lightly loaded applications result in extremely low collision rates. Using collision avoidance instead, these networks save cost and complexity in their bus interface boards. In collision avoidance, message collisions get detected by the sending and receiving nodes' circuits. In a common detection technique, the receiving node delivers an acknowledgement signal back to the sender. Should the sender not receive the acknowledgement, it usually retransmits until successful.

TRANSMISSION MEDIA

Transmission media fall into three fairly distinct cost/performance categories:

a) TWISTED PAIR
- low cost, easy to install
- bandwidth limitations
- distance limitations (less than 1 mile at 256 Kbps)

b) COAX CABLE
- heavy shielded cable
- high bandwidth (greater than 10 Mbps)
- Good noise immunity

c) FIBER OPTICS
- very high data rates (greater than 100 Mbps)
- Excellent noise immunity
- difficult to make tap connections

Twisted pair provides an excellent media for integrating voice and data through a digital PBX system. This type of PBX based network allows for the low cost interconnection of a large number of workstations (terminals/ personal computers), all through existing twisted pair phone wiring. Fiber optics, although currently exhibiting the highest transmission medium price, possesses inherent point to point performance capabilities that outclass all other transmission media. Its key characteristics include virtually unlimited bandwidth, high gigabit per second data transmission speeds, insensitivity to electromagnetic interference, complete electrical ground isolation between transmitter and receiver, high voltage isolation, nonelectronic radiation, small size and light weight. Despite these impressive properties, though, fiber optics is still too costly for multitapped connections. Compared with low cost twisted pair wiring, shielded coaxial cable's moderate cost, high performance, ready availability, easy configurability and wide bandwidth make it the most popular transmission medium for interconnecting local networks. Because it maintains low level

capacitance in lengths to several miles, coax allows high megabit per second data rates without signal regeneration, echoes or distortion. These features reflect a field proven technology with more than 20 years of use in data communication networks and CATV applications.

SIGNALING TECHNIQUES

When using a coax based local area network, two primary signaling techniques are used 1) BASEBAND and 2) BROADBAND. A baseband network uses purely digital transmission techniques and usually has a maximum capacity of approximately 10 Mbps. The distance of a baseband network is usually limited to two kilometers, and the network is generally used for data transmission only. A broadband network uses analog transmission techniques, and can accomodate upto 400 Mbps of information. This information is carried through a broadband network by creating several logical cables within the one physical coax cable (Fig 4). A broadband network can handle multimode transmission (data/voice/video) simultaneously and extend for distances up to twenty miles. Broadband's principle advantage lies in its immense information handling capacity: One broadband cabled local network accomodates many thousands of connected nodes, handling nearly all the word, data, voice, video, and image communications generated by a busy high traffic system. These communications might include broadcast and closed circuit TV, video surveillance, telephone calls, facsimile, word messages, and data transactions. However broadband networks have several major shortcomings. Baseband networks can make use of clamping tap connections. When a clamping tap is used, systems can be easily and quickly connected to or disconnected from a baseband coax at any location without disturbing network operations. Currently, connections to a broadband coax must be inline. To make an inline connection, one must shutdown the network in order to sever the cable. (When installing a broadband network one should try to plan ahead by installing extra interfaces so that disruption to network services can be minimized. In addition to requiring network and station interfaces, broadband networks use expensive fixed frequency or frequency agile modems costing $500 to $1200. Not only does this double the broadband interface cost, but tunable RF modems prove difficult to check, maintain, and adjust because they are usually installed behind walls and above ceilings. Broadband networks also rely on a central transmission or head end facility in a single cable network. This facility acts as the network's technical control center. It filters incoming RF signals from multidropped sending nodes and retransmits them at higher frequencies to receiving nodes. This central transmission facility represents a point that, if it fails, can deactivate the entire network. One other major shortcoming of a broadband network is that there are no widely accepted standards in the industry for interfacing equipment to these networks. Several vendors have endorsed the IEEE 802 standard for baseband networks, and this will allow the implementor of an IEEE 802 network to select from a wider variety of equipment that will be able interface to his network. Selecting a broadband vs. a baseband network will again largley depend on your application. However, for applications where the primary use of the network is for data only, a baseband network provides significant cost/per- formance advantages.

FUTURE ISSUES FOR LOCAL AREA NETWORKS

NETWORK MANAGEMENT - As larger numbers of systems connect to the LAN, having the tools to allow effective resource management and planning will become increasingly important.

SECURITY - Most organizations want to protect the confidentiality of traffic flowing over the network. In some ways a LAN provides more security than conventional office communications, such as phone or interoffice mail. Converting voice and paper information to electronic signals makes them less accessible to the ordinary office worker. On the other hand, a communications network is a powerful tool in the hands of a sophisticated information thief or saboteur. Vendor's will have to develop new security techiques to prevent misuse of network.

TECHNOLOGY IMPROVEMENTS - The encapsulation of LAN communication systems onto VLSI silicon chips will contribute significantly to the promotion and general acceptance of LAN facilities. THe VLSI will reduce the LAN component prices and enforce a level of LAN standardization through sales of volume chips.

BRIDGES AND GATEWAYS - The creation and use of LAN facilities will demand the development of mechanisms to 1) interconnect various LAN's to each other and 2) interconnect LAN's to long haul networks to allow long distance informational exchange and resource sharing.

HYBRID NETWORKS - For many organizations, no one network topology, access method, or media may suit all of the user's needs. In these organizations hybrid networks will evolve. The management of information across these hybrid networks will continue to be a challenge during the coming years.

SUMMARY

The creation and effective implementation of a local area network is the key to unlock the full potential of the "information age". A LAN should be used to bind the distributed systems within an organization into a unified resource. The effectiveness of this total resource will then be measured by added capability and the degree of coherence that it achieves. This is turn will depend on the care and foresight put into the design of the network and the development of standards for interworking of systems at all levels.

REFERENCES

1) Antony, P., "Local Area Networks: A Tutorial Overview", HPIUG 1982 PROCEEDINGS - COPENHAGEN, October 1982.

1) Heard, K., Local Area Networks, Gartner Group Special Report, February 1982.

2) Hopkins, G. and Meisner, N., "Choosing between broadband and baseband local networks," MINI MICRO SYSTEMS, June 1982, pp 265-274

3) Kinnucan, P., "Local Networks Battle for Billion Dollar Market," HIGH TECHNOLOGY, November/December 1981, pp 64-72

4) Kotelly, G., "Local Area Networks - Technology", EDN, February 17, 1982, pp 109-119

New Trends in Workstation I/O

Jim Beetem
Catherine Smith

1. Introduction

The purpose of this paper is to discuss what we in the Data Comm lab feel are the major trends in terminal and workstation I/O, some new tools we have for dealing with newer devices and connection methods, and how programmers and system managers can best connect and utilize these new devices. We are using the term workstation vs. terminal to emphasize the fact that it is not only terminals which are connected over serial communication lines, but also personal computers, printers, plotters, and black boxes.

The paper is organized into four sections. The first section discusses the current connection methods and how we see these methods evolving. The second section mentions features we think will be added, deleted or changed from current terminal drivers. Its emphasis is on features in the point-to-point terminal drivers, since those features are the ones which have evolved more (vs. being planned). The third section discusses trends in support: tools which we are providing for the user, programmer, and system manager. This secion also is concerned with point-to-point connections only. The last section offers some hints and suggestions on how best to program and/or configure non-traditional devices through terminal drivers on the 3000.

2. Current connections & trends

The current connection methods supported by HP for workstations are: point-to-point (via the ADCC and ATP terminal drivers), multipoint (through multipoint terminals and printers or the 2333A cluster controller), X.25 (through PADs or the 2334A cluster controller). There are other methods available from other vendors, some of which work very well. However since these connection methods have not been tested by HP, they are not supported. These other connection methods will not be discussed.

Workstation communication has evolved from teletype communication. Originally this involved infrequent data transmissions, one character at a time at low line speeds. Speeds have gotten faster and workstations much more intelligent which has resulted in new uses and needs for workstation I/O. Applications which do large transfers at line speed such as block mode, downloading, and file transfer are more prevelant. Also, due to the increased amount of data communication in general, the topology of workstation connections has changed. The old choices were either hardwired terminals located close to the system, or terminals connected via modems. A third major change is that the devices being connected over serial lines are different. Instead of terminals, we are finding more personal computers, printers, plotters, and other non-traditional devices. All three of these changes, speed increases, topology changes, and types of devices connected, have put a higher importance on data integrity.

2.1 Point-to-point connections

Many point-to-point terminals connected to a system is the default connection method. Since it is the default connection, almost any application program written for 3000 terminals will run on point-to-point terminals. Connection methods for point-to-point terminals are numerous and usually inexpensive. They can be connected through modems or hardwired. With RS-422 and fiber optics multiplexors they can be hardwired at great distance from the system. Modems for async communication are inexpensive. Another big advantage of

point-to-point workstations is that if something goes wrong (because of the terminal, the application, or the user) it only affects the one terminal. Serial printer support is available for point-to-point connections.

Point-to-point connections have several disadvantages. Since they are point-to-point, there is a separate cable for each device. This requires lots of junction panel space at the system and a lot of cost for the cables and installation. The cost problem is more acute if there are a lot of remote workstations, since there must be two modems per workstation. Also point-to-point terminals have no data integrity since there is no error checking other than parity. This is becoming a bigger problem due to the increase in 8-bit terminal types for foreign language and file transfer applications. The only way that point-to-point devices may be used on other than the primary system is via DSN/DS. Also since point-to-point connections are so flexible, there is a larger variety of applications/device running over point-to-point which makes support difficult since there are more corner cases.

One of the new connection methods for point-to-point workstations is through PBXs. The advantage that this connection method has is that the wiring is already in place for voice communication and so can be used for data communication. It also allows for connection to multiple systems and so enables pooling of systems and system resources (such as modem pools). All applications and devices supported by the point-to-point controllers are supported over PBX connections.

Unfortunately, PBX connections are currently high cost compared to traditional point-to-point connections. PBXs have fewer numbers of switch points than logical connections, so if everyone picked up their phone at once, not everyone would get a dial tone. This means that users who don't disconnect the PBX connection after they have finished communicating with the system, are wasting a valuable resource. PBX connections also have no data integrity other than parity, although the error rate is very low.

2.2 Multipoint terminals

Our current solution for solving some of the deficiencies of point-to-point connections has been multipoint. Since multipoint devices are connected on a single line, cabling costs are reduced. This reduction is especially noticable for remote connection. Only one modem on the system side, and one modem for each drop of devices are needed. Multipoint connections can be either asynchronous or synchronous. Synchronous connections are used for remote connections so that the faster synchronous modems may be used. Asynchronous connections are used for local connections via the factory data link. Not only does the data link provide better imunity to noise, but multipoint also has excellent data integrity via CRC checking.

Multipoint also has its disadvantages. Since it does use CRC, there is more overhead than for point-to-point terminals. Also multipoint terminals are inherently block mode, the system does not receive any characters until the user hits enter. Because the command interpreter does not know how to talk to block mode terminals, the multipoint software translates the data to look like it came from a point-to-point terminal. This extra software also adds extra overhead for communicating with multipoint terminals. When a block mode interface like V/3000 is used, no conversion from character mode to block mode is needed, so multipoint runs more effieciently and overhead is reduced.

Since multipoint terminals are inherently block mode, some applications must be changed to run (or to run well) on multipoint terminals. Since multipoint terminals share a single cable, the response time and throughput for one terminal depends on how much of the line bandwidth other devices are using. So what one device does affects more than just itself, it affects all other devices on the line.

Another multipoint connection is via the HP 2333A cluster controller. This allows the user to connect point-to-point terminals to the controller which is connected via multipoint to the system. CRC checking is used to talk to the controller, so data integrity is achieved. Since point-to-point terminals are connected to the cluster controller, any application which can run as terminal type 10 can run on a terminal connected to the 2333A. The 2333A cluster controller allows a wider range of workstations to be connected to the multipoint line.

Workstations connected via the 2333A have some of the same disadvantages as multipoint devices. There is some software overhead on the system to communicate with the 2333A as well as the workstation and all workstations connected to the 2333A are sharing the multipoint line. Also although any software used to run terminal type 10 terminals can be run through the 2333A, no other terminal types are supported and some applications (such as keystroke interaction applications) will not run well.

2.3 Terminals connected over X.25

There are two ways which terminals can be connected over an X.25 packet switched network. The first is via a public PAD (packet assembler disassembler). To use a PAD, the user calls up the local PAD (a local phone call) and then gives the PAD commands to "dial" a

system. This is a very cheap method for long distance communication (if the system end has other uses for the X.25 connection) since the X.25 price is not tied to distance. Instead of being tied to distance, prices for X.25 connections have some fixed cost for connection to the network (small for connections to PADs) and then a very small cost per packet. This connection also is beneficial if the terminal user uses many systems on the network.

On the minus side, PAD terminals have many of the same disadvantages as multipoint terminals. Communication must work its way through the network and so response time is often slow. The response time also depends on the application. An application which runs poorly over multipoint is likely to be worse over an X.25 network. Also all features are not supported over a PAD terminal. Currently the only block mode which can be used is via V/3000. Since terminals are not permanently connected to a system, there is no way for the system to FOPEN a PAD terminal. Since printers can only be FOPENed (they can't logon), printers are not supported through PAD connections.

The second way in which terminals can be connected to an X.25 network is via the HP 2334A cluster controller. This connections method has many of the same advantages and disadvantages as a terminal connected to a public PAD.

Workstations connected to the HP 2334A cluster controller provide a cheap long distance solution if there are many workstations connected at one location. There is an additional expense since the 2334A is connected to the X.25 network as a system, not a PAD. This adds a larger monthly expense in this country, but not much more in Europe. Printers and terminals may be FOPENed since the 2334A cluster controller is connected as a system (a permanent connection).

Terminals connected through the 2334A have the same response time limitations as do terminals connected through a public PAD since data must work its way through the network and the system software. As with PAD terminals, V/3000 is the only block mode supported for terminals connected. The cost of 2334A terminals over PAD terminals only becomes favorable to 2334A terminals if there are a lot of terminals at one location.

3. New Features

As mentioned earlier, there are several trends apparent in the terminal I/O area which must be reflected in the design of future HP systems. First, to handle the demands of personal computers and intelligent workstations, new systems will need I/O facilities that handle data in large blocks with "perfect" data integrity. Second, the traditional asynchronous terminal I/O port will be used to connect an increasing variety of devices, each requiring a different set of characteristics. Many of these devices will not be compatible with HP terminals. Third, as the price of high qulity, high speed printers decreases, HP 3000 customers wish to move these devices closer to the user's work area. Terminal I/O ports (either point-to-point or multipoint) provide the only cost effective solution to this need. Fourth, some of the facilities that are now widely used; many of these facilities will not be implemented on future I/O

In the following section the discussion is aimed at point-to-point terminal connections. Other terminal connections either have solved the problems or the problem never existed for that connection method.

3.1 A Reliable Link to Work Stations

Personal computers and intelligent workstations often move entire files of data to or from the HP 3000, manipulate the contents and move it back. Their primary concern is that a large amount of data be moved quickly and accurately between the two machines. This is a new requirement in the HP 3000 Terminal I/O environment.

The current terminal I/O interface on the HP 3000 was designed for use with simple "TTY-compatible" terminals. It is intended to transfer one character at a time at human typing speeds inbound and to print data on the display at human reading or scanning speeds. HP has enhanced this basic design to handle VPLUS and HPWORD block mode terminals with reasonable reliability and efficiency. However, the new devices demand much higher thruputs and require more efficient operation to avoid degrading system performance.

The primary source of data integrity in the existing design is inherent in the physical facilities. Within their speed/distance limitations, the RS-232-C and RS-422 standards offer a very high quality link to local area users. Where telephone lines must be used, only parity checking is availale. In the future, HP is committed to using the 8th bit for data not parity, so even this facility will not be available.

The solution to these requirements is an efficient link protocol. It should allow movement of masses of data at close to the capacity of the data comm line connecting the machine. The protocol should use an error detecting mechanism like Cyclical Redundancy Checking (CRC) and automatically trigger retransmission of blocks containing a error. LINK/100, a product that transfers data between the HP 3000 and the HP 100 series of personal computers is a step toward an efficient and reliable workstation link.

3.2 Broader "Connectivity"

On current HP 3000 systems, each terminal I/O port must be configured as a specific "terminal type". The "terminal type" describes the set of facilities that are operational for that port. Any terminal connected to that port must be compatible with those facilities. For example, any terminal connected to a port with the ENQ/ACK handshake enabled must be able to perform that protocol.

HP provides point-to-point terminal types that allow our software packages to operate HP terminals properly. However, customers developing their own software or wishing to connect non-HP devices to the HP 3000 often have problems finding a terminal type that exactly fills their needs.

The solution to this problem is a new HP product, the Work Station Configurator (WSC). This product allows customers to specify their own terminal types as files that reside on the disc. These user-defined terminal types may be configured as the default terminal type in SYSDUMP, specified as Log-on terminal type in the :HELLO command, or changed programmatically at run time. Users cannot add new capabilities to the existing terminal I/O interface, but they can selectively enable or disable almost all of the available facilities.

The new terminal types that can be created by this product (and by the associated enhancements to MPE) allow users with unusual needs to tailor the actions of the terminal I/O interface to their specific device. For example, none of the HP-supplied terminal types allow V/3000 to by used without the ENQ/ACK flow control handshake. A user-defined terminal type can eliminate this restriction, allowing more efficient operation of statistical multiplexers. For printers, the FOPEN-time Initialization string that is written to the device may re-defined, allowing users to initialize the device to any configuration of margins, character size, paper size, pitch, etc that the device supports. Since the default terminal type in the I/O configuration can be temporarily changed by the ;ENV= clause in the :FILE command, subsequent jobs on a spooled device are not affected by the unusual requirements of a single job.

As new capabilities are added to the terminal I/O interface, control of the capability will generally be available to users thru the WSC facility.

This product is intended for knowledgeable users. The WSC allows users to specify any set of the available features. Many of the sets would not be satisfactory for general use. There is no method that the WSC product can use to detect possible problems. For example, a user-defined terminal type with all flow control mechanisms disabled is likely to suffer data overruns at high line speeds. A terminal type that sets the ENQ/ACK handshake block size to 10 characters will operate inefficiently, especially over statistical multiplexers. The detection and prevention of such problems with user-defined terminal types is the user's responsibility.

3.3 Improved Printer Support

As the price of high quality, high speed printers continues to decline, more and more HP 3000 customers want to move these devices into the user's work area. The only economical method of connecting these devices to the HP 3000 is via one of the serial I/O facilities.

Between the point-to-point I/O port and the device, only the physical quality of the data comm line is available to insure data integrity, so that connection of printers must be limited to the distance supported by the Level 1 standard, either RS-232-C or RS-422 (ATP only). Special care must be taken in electrically noisy environments to prevent data errors.

The only exceptions to this restriction are devices that operate as terminal types 19 and 21. These are 7 bit devices that always operate with parity checking enabled, allowing them to detect but not correct data transmission errors. Multipoint printers obtain data integrity through the CRC checking inherent in the multipoint protocol.

HP now supports terminal type 18 for connection of printers that depend only upon the quality of the data comm line for data integrity. The use of terminal type 18 devices across phone lines with modems is NOT supported, since their quality is never high. In many cases there is one error for each one hundred characters transmitted.

3.4 Obsolete Features Not Expected To Be Supported

Many of the features of the existing terminal I/O interface were added to support a specific device; when the device was no longer sold and the need for the feature went away, the feature stayed on. Many of the current features were intended to support simple TTY terminals

[redacted]

tributor to the data overrun problems experienced by the ADCC at high line speeds. Most of these features will not be supported on the next generation of terminal interface products.

a. Control of Echo from the terminal: typing (ESC) ; and (ESC) : allow the user to enable or disable the input echo facility. Scanning the input stream for the (ESC) character is a major problem. It is very unlikely to be available in the future. Programmatic control will be available; if sufficient demand is present, an MPE command like the :SPEED command could be implemented.

b. Line Feed triggers output of Carriage Return: to facilitate input on certain devices, the HP 3000 will echo two characters, Line Feed and Carriage Return, in response to a single Line Feed character. This feature also requires scanning the input data stream.

c. Substituting Line Feed for Form Feed in Output streams: this feature handled a device that didn't recognize the Form Feed character. That device (the HP 2615) has not be sold since 1974; it will not be supported on future

the data stream, some other flow ocol must be used. Defining and g such a protocol must be done by the user.

f. Input Data Saved over a Break: at present, a user can type an MPE command, type BREAK, type some other MPE commands (like :FILE), type :RESUME, then type Return and the first MPE command is now executed. It was saved during all of the subsequent transactions. This facility is largely unknown by users and probably will not by implemented on future interfaces.

g. TTY Delays: early terminals were not able to process all characters in "real time" and had no buffering; this required a short delay following these characters. The HP 3000 inserts a short delay after Carriage Return, Line Feed and Form Feed for some terminal types. Support of this feature requires scanning the output data stream, adding to system overhead and slowing the output data rate. This feature will not be supported on future controllers.

h. Slow Line Speeds: current terminal interfaces support most of the following line speeds: 110, 150, 300, 600, 1200, 2400, 4800, 9600 and 19200 bits per second (bps). Future interfaces are unlikely to support speeds of 110 and

errors; instead, the state of the single terminal port affected by the error is dumped to an extra data segment, the port is marked "Broken" in the Device Information Table (DIT), a message is printed on the system console and operation of the port is suspended. As a result, that single port is "broken", but all of the remaining ports can continue normal operation.

To "repair" the broken port, the System Manager or Operator should first use the DUMP facility in TERMDSM to print the data dumped to the extra data segment and then mail it with a software service request to his local HP Service office. (This report allows the factory to find and fix the software problem.) Then, the broken port can be RESET, aborting the user's session, and the user can log on again to resume processing. In a few cases, the RESET fails. This most often happens because the "critical" bit is set for this process. When this happens, a WARMSTART is required to clear the port.

The DIAGnostic facilities of TERMDSM allows some hardware diagnostics to run on-line, affecting only as many users of the system as absolutely necessary. (No on-line diagnostics exist for the ADCC hardware.) The loopback capabilities allow troubleshooting of intermittent data loss or cabling problems.

formation can be very valuable to help determine why a port or process appears to be "hung".

There are several contributions that this tool makes. First, it allows HP to obtain data to fix software problems without serious inconvience to our customers. Second, it allows customers to fix simple software problems themselves, without waiting for an HP Customer Engineer to come to the site. Third, the diagnostic and display facilities allow troubleshooting of terminal I/O problems in either HP hardware or software or in customer applications without bringing down the entire system.

This tool is typical of the approach HP expects to take in future products, where facilities are provided that allow the customer to do more of the system support work, lowering the cost of support by minimizing the need for expensive on-site visits by HP support personnel.

4.2 New Point-to-point Workstation I/O Manual

The new Point-to-Point Workstation I/O Reference Manual (30000-90250) will be available with MPE V/E. It expands greatly the previously available information on the use of the MPE intrinsics to perform terminal I/O.

Another problem we had been seeing a lot of is devices which do not fit any of our supplied terminal types. If a user wanted to connect some black box that didn't do ENQ/ACK handshake but did need the DC1 read trigger, there was no way to do it. This last problem has been solved with the new Work Station Configurator user definable terminal types.

Jim Beetem is a project manager in IND's Data Comm lab where he was responsible for the ATP product and the INP software. His undergraduate degree is in Mechanical Engineering from MIT; he has an MBA from Stanford where he has done graduate work in Computer Science. Jim is married and has two small children, who occupy most of his "free" time.

Catherine Smith is also a project manager in IND's Data Comm lab. She joined HP in 1978 after receiving her BS degree in Electrical Engineering and Computer Science. She has worked on software for multipoint terminals, point-to-point terminals, and the 2333A cluster controller. Catherine, her husband, and her teenage son spend as much time as possible skiing.

Performance Characterization of Disc Caching: A Case Study

Becky McBride & Sam Boles

Hewlett-Packard

Synopsis

Disc Caching is one of the most significant performance enhancements to be introduced in the HP3000 line. The basic theory is clearly sound, but even in the conceptualizing stage the development team knew that empirical validation was essential before announcement of the product.

As soon as the code stabilized after integration, the Cupertino Systems Performance team moved in to subject it to an intensive 4-month performance characterization exercise.

The performance team analyzed existing benchmark test sets and selected a range of database and sequential file tests. This approach gave a real-world flavor to the test sets, and leveraged on the substantial engineering resources already invested in performance benchmarking.

The team then created an environment in which they emulated terminal load by driving the system under test with terminal messages generated by another computer direct-connected to the system under test. This enabled not just a measurement of resource utilization on the system under test, but also a measurement of response times and transaction throughput time at the individual "terminal" level. The automation of the terminal load also enabled precise repeatability and exactly controlled think time and typing speed -- all critical to the measurement of sensitivity to memory pressure, disc configuration changes, number of concurrent sessions and other variables.

The joint effort of the development team and the performance team resulted in a definitive performance profile of the product prior to manufacturing release.

Background: The Other 2 1/2 Rule and the Bottom Line

The Other 2 1/2 Rule

Back in the Dinosaur Days of computers - 20 years or so ago - somebody at the Pentagon came up with the "2 1/2 Rule." They had had a lot of experience with the giant, monolithic systems contracts for the Defense Department. The Rule went something like this: Big computer systems take 2 times as long to build as planned, cost 2 times as much, and have 1/2 the functions and performance.

Any of us with combat experience building big systems (or even little ones) can empathize with the 2 1/2 Rule. We might debate the numbers (they may be too optimistic), but the message is painfully true.

There's another 2 1/2 Rule in the computer industry. It's a little more positive. It says: Every 2 years deliver twice the performance or cut the price in half -- or do both.

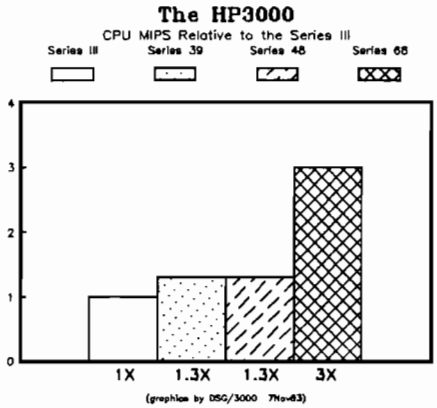
The 2 1/2 Rule and the HP3000

We've all seen the good 2 1/2 Rule in action in the HP3000 family. It's a growing family in performance and functionality and it's competitive price-wise in an industry whose amazing productivity contribution is surpassed only by it's dramatic price competitiveness.

A Few Surprises

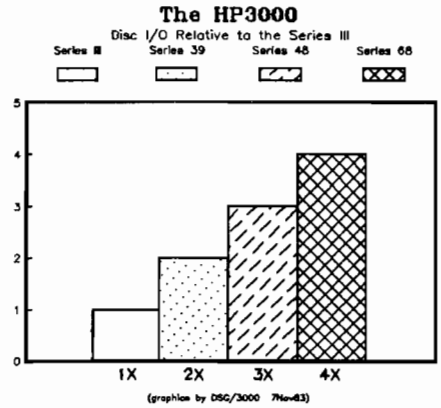
We've seen some impressive advances in processing power, main memory capacity, I/O rates and other system components. And we've seen performance gains. But we've had a few surprises.

Let's take a look at what's happened.



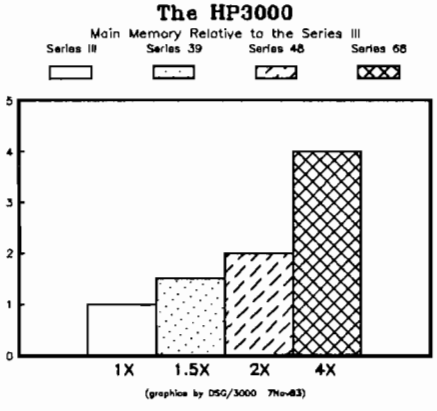
. . . With MIPS Like These . . . .

In the area of CPU (Central Processing Unit) power, we've seen the MIPS (Million Instructions Per Second) grow by more than a 3X factor as we moved from the Series III (about 0.3 MIPS) to the Series 64 and 68 (1.0 MIPS).



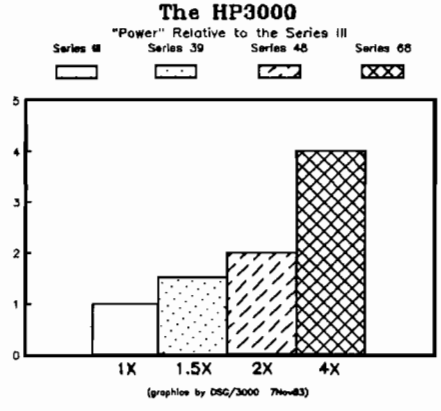
. . . and Disc I/O Rates Like These . . . .

From the 30 or 40 disc I/O's per second on the Series III we've grown to the 120-160 (and higher) range on the Series 64 and 68.



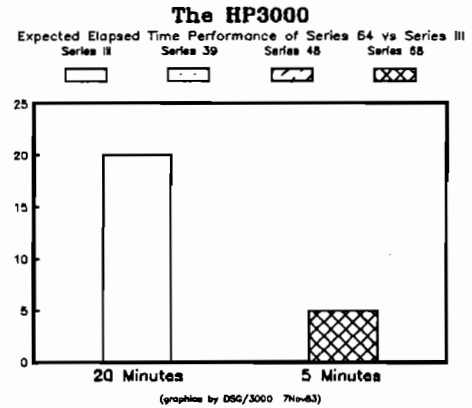
. . . and Main Memory Like This . . . .

The main memory capacity of 2 Megabytes on the Series III has grown 4X to 8 Megabytes on the Series 64 and 68.



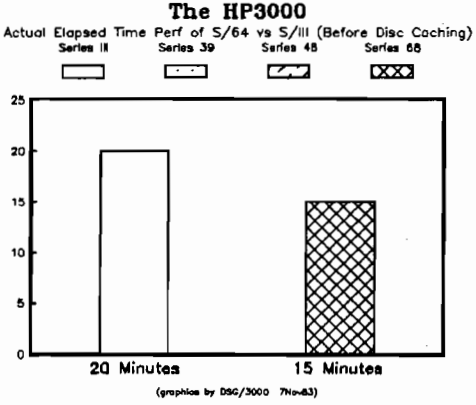
. . . You Get a Power Curve Like This . . . .

When you put these components together you get some entropy (like CPU consumption from the overhead of moving more I/O load) but a lot of synergy (like more supported terminals to utilize the greater disc capacity) so that your data processing "power" looks something like at least a 4X gain.



. . . So We Expected This . . . .

With 4X the data processing power, we thought we could take 20 minutes' worth of work on the Series III and do it in 5 minutes on the Series 64.



. . . But We Got This . . . .

When we took the 20 minutes of work from the Series III and ported it very carefully, 1-for-1, to the 64 and ran it there, we found we got only about a 25% improvement in performance.

The Bottom Line

Somehow *it wasn't getting there.*

If It Hasn't Got It There, It Hasn't Got It

We knew from our unit tests of individual components that we had the raw horsepower in the box to do better than that. But in the systems test, benchmarking customer data and customer programs, it wasn't making it to the Bottom Line.

The Autopsy of Our Shattered Dream

We started digging into the problem. We looked at some of the stats from the instrumentation we usually enable when running a benchmark. We found that we had the *CPU busy less than 50%* of the time and paused for I/O over 40% of the time. We were idling at a 75 nanosecond rate -- a very *fast idle*.

We looked at main memory utilization. Of the 8 Mbytes we had configured we were using between 5 and 6 Mbytes, *with 2 Mbytes standing idle.*

We looked at disc I/O. We were doing *only 50 I/O's a second* on a system where our unit tests had sustained better than *3 times that load.*

We found that we were getting a *process stop about 40 times a second from impedes on the database control block* (single-threaded to assure data integrity and correct linkage.)

For a number of reasons, the majority of which were directly or indirectly related to disc I/O, we were not realizing the performance potential of the Series 64 when we did a simplistic migration of the typical HP3000 job mix.

The Birth of Disc Caching

Enter the Busch-Kondoff Team

About that time, John Busch, architect of the MPE Kernel, and Al Kondoff, a disc I/O specialist in the MPE lab, focused on the problem.

The Memory Hierarchy

They looked at the time it takes to get data from main memory. That's about 400 nanoseconds. They looked at the time it takes to get data from disc. That's about 40 milliseconds. *A 10000X difference.*

The Typical HP3000 I/O Profile

They did extensive sampling of a wide range of HP3000 job mixes. They built instrumentation to measure and characterize fetch sizes, temporal and spatial locality, read:write ratios.

They built an I/O simulator and analytic model to evaluate design alternatives and emerged with a new idea.

The Mechanics

John and Al devised a scheme that would use the 1-2 Mbytes of idle main memory on the 64 to store large blocks of disc data that were local to data specifically requested by the user. If you asked for record #1, they'd read in record #1 and the next 20 Kbytes or so of data, on the assumption that you'd ask for record #2 soon and other records in that vicinity. As these disc domains get cached into main memory, when you asked for record #2, you'd get it in *4 milliseconds rather than 40 milliseconds. That's 10 times faster.*

From the analysis of real-world I/O traffic patterns, it looked as though you would get a hit in the disc data cached into memory about 70% of the time - more than enough to make the several millisecond disc cache overhead a high-yield investment.

Not a Pyrrhic Victory

The Cache management was integrated at the Kernel level so that disc cache was competing for main memory via the same approximate LRU (Least Recently Used) algorithm that manages code, stack and extra data segments. This enables a dynamic sensitivity to the varying resource needs of the system at the global level. That way you don't sacrifice some other important sector of performance just to peak your disc I/O's.

When John told us about his simulation and analytic model results, we were very interested. It sounded as though he were on to something. Something that we really needed in the Systems Performance Center.

The Systems Performance Center

In Cupertino, California, the location of the key HP3000 R&D Lab, the Systems Performance Center runs customer benchmarks and does performance characterization of HP3000 hardware and software products.

A typical scenario of the Performance Center operation goes like this: A customer has a Series III and has outgrown it. He wants to see what response time and transaction throughput rates he would get with a 64. He and his SE and Performance Specialist in the field analyze his job mix and select programs and data for a benchmark. (They usually expand the load in order to do a little *ad hoc* capacity planning along with the benchmark.)

The field/user team then makes reservations in Cupertino and joins the System Performance Team there to run the customer's programs and data on equipment in the Performance Center configured to reflect what has been proposed as a solution.

The key element in this exercise is its *real-world connection*: it's customer programs running customer transactions against customer databases, in many cases, ported directly from the customer's production environment.

On-Line Dimension

A major contribution of the 3000 is to bring the on-line interactive dimension to the world of commercial data processing. With the customer still on the phone you can see that you have a hundred widgets in stock that you will ship him today, and 500 more are coming out of QA tomorrow that you will Federal Express to him. He says "Go", you update the inventory (so nobody else gets promised the same 100 widgets) and launch the order. Now. *In almost real-time.*

This is the world of the HP 3000. And this is what customers want to benchmark. But with the Series 6X, they want to see the response time they will get with a hundred terminals going at the same time. And what will happen to response time if they add a Meg of Memory? And what will happen to response time if they add another 7935 disc drive? Another GIC? Another ....

Benchmarking Tools

Well, how do you get a hundred terminals, hook them up, get a hundred operators, get them to enter data with the right think time and typing speeds, make errors only when you want them to, measure the response time at each terminal for each transaction (down to a hundredth of a second), get them all to

start at a specific time, stop at a specific time, not sneeze or stop for coffee except when you want them to, then do the whole thing over again exactly the same way in every detail after you have changed the disc configuration for the next test (which will be about midnight, at which time you want all hundred operators and timers and terminals and files on the mark, ready to go -- and it's Saturday night)?

How?

No how.

You can't do it.

Except maybe with a computer. Like an HP 3000, for instance.

TEPE

You take a terminal emulator like HP's TEPE (Terminal Emulator and Performance Evaluator) and write "scripts" for it. You tell the program the think times, typing speeds, baud rates, programs to run, answers to give to the program you're running. You build a hundred of these, varying whatever you want to vary, then compile them into a form that the computer can use.

Then you take a couple of 44's and configure them with a special terminal driver. You get some special TEPE cables that hook right onto the ADCC (terminal handler) edge connector on one end and onto the 3-pin RS232 ATP connector on the other end at the Series 64 junction panel.

The System Under Test

On the 64 (the System Under Test) you load your application programs and databases and other files. On the 44's you start up the TEPE monitor, tell it what scripts you want to run with what start times.

Then you launch. Each 44 takes 50 scripts, analyzes the parameters and sends an RS232 bit serial message down the wire to the 64. By the time they hit the 100 ATP connectors on the 64 junction panel it looks just like what would have arrived if you and I and the other 98 operators had been sitting there banging real terminals. Only it's *exactly repeatable. And controllable. And changeable.* And response time is measurable at a fraction of a second. And it's *economically and ergonomically feasible.*

Emulation

This is *emulation. Not simulation.* It's real RS232 messages coming over real wires to a real System Under Test. This is how you execute and measure large configurations of on-line systems.

While you're running the benchmarks, the TEPE monitor on the driver 44's is measuring the inquiry-to-first-response response time on each "terminal" so at the end of the benchmark you can get mean resonse time, min, max and standard deviation for individual terminals or for the whole load, for the entire test or for a selected steady-state window.

Over on the System Under Test, you enable instrumentation like OPT (On-Line Performance Tool), to get CPU, main memory and disc I/O statistics with variable granularity: global total, global by interval, logical device level. You can get system table utilization, working sets by process, and other components of a wide range of performance statistics.

Disc Caching Into Battle

It was into this arena that Disc Caching came to prove itself. This was *real-world programs, real-world data, real-world loadings, real-word response time requirements, right out of real-world customer production systems* around the real world. Where it deviated from the real-world was where customers injected worst cases (eg: zero think time) to stress-test the system.

Now remember where we performance engineers were coming from: we'd examined John's and Al's design. They had done their homework. Their design was esthetically beautiful. Their implementation showed the consummate art of skilled craftsmen. And we wanted very much to have something that would enable the 64 to do the job it was built to do.

But in the Performance Center *the real-world Bottom Line is what counts. If it hasn't got it there, it hasn't got it.*

The First Round

We went to the Benchmark Library to get some of the tougher customer tests we had run. We selected one with a 56 data-set IMAGE database and a couple of KSAM files. We set up for 21 terminals and gave it a shot. We first ran with a pre-Cache Q MIT (the official current MIT at that point in time) to calibrate a base line. One of the design criteria John and Al had set for themselves was not to degrade performance on a system where Disc Caching was not enabled.

We then restored the database, reset the system, updated to the Cached Q-MIT and ran the same 21-terminal test with Caching disabled. We compared the numbers (overall mean response time inquiry-to-first response) and other stats of finer granularity and found no degradation due to Caching (in fact, there was a small improvement from some minor optimization John and Al had done while they had MPE "open on the table.")

The Caching Parallel: Some Gain

We then re-staged (reloaded the databases, etc), enabled Caching on all LDEV's and repeated the 21 terminal-test. We got about a 20% performance improvement. That was better than nothing, but like tapioca -- *good, but not exiting.*

As with the other tests we were running with OPT and MPEDCP (MPE Data Collection Program) enabled with

logging at 60-second intervals. This, in addition to the TEPE response time stats as seen at each "terminal", gave us a wide range of granularity in monitoring this extensive instrumentation.

John and Al pored over the stats, and proceeded to optimize the code. We staged for the next shot and they gave us a new build. We ran with a 25% gain this time.

The queue length at the LDEV level showed that we didn't have enough load with only 21 terminals to get the full "Cache Effect." We went to 42 terminals, then to 63 and started to get 50% (2:1 response time reduction) improvement and better.

Stage, Shoot, Tune, Build

We worked into Saturday night with a Stage-Shoot-Tune-Build routine, tweaking things like the cache-flush algorithm and fetch size. Most iterations picked up a few points. Some were disasters that we had to undo. We were tired but encouraged.

We came back Sunday and went into the Stage-Shoot-Tune-Build mode again. We pulled in some other benchmarks from the library. We ran some sequential file processing. With the larger fetch sizes and higher cache hit rates we got some very impressive results.

As the week wore on, we widened the range of benchmarks. We took the 56-dataset IMAGE benchmark with 63 terminals doing transaction processing and added 38 developmental (COBOL, FORTRAN, BASIC edit, compile, test) sessions. Here we had a hybrid benchmark with 101 terminals accessing IMAGE, KSAM, and sequential files.

We ran with Caching disabled. Mean Response Time (Inquiry to First Response): 0.92 seconds.

3:1 Improvement

We restaged, enabled Caching and ran again. Mean Response Time: 0.31 seconds. A 3:1 improvement. We knew we were on to something big.

After two weeks we cleaned up the debris, sifted through our data, had an engineering review by the development team and the full Performance Engineering team. We decided to integrate a Disc Caching parallel phase with every benchmark coming through the Performance Center.

Looking for the Holes

We were getting very impressive results where we had a good fit:

20-40% available CPU
1-2 Mb available main memory
Average file locality on disc
Average (3-4:1) Read:Write ratio
Heavy disc I/O load

We had found that there were conditions where Disc Caching not only didn't help but actually degraded performance. We focused on defining those conditions, so as to minimize the occurrence of misfits.

We found these conditions where Disc Caching had marginal or negative effect:

CPU busy >65% of the time
Memory Pressure > 10%
Low hit ratio in a 4 Kb IMAGE or 24 Kb sequential fetch size
Block size about equal to Caching fetch size

Very Good Becomes Better

As we gave Disc Caching more and more real-world exposure John and Al added some features to make Caching more flexible.

In addition to the STARTCACHE & STOPCACHE MPE commands at the LDEV level, they built a CACHECONTROL facility that would allow user specification of fetch sizes for both sequential and random files. This enables some remedy to situations with a locality problem. The fetch size can be expanded to get better hit rates; or, if that doesn't remedy the problem, the fetch size can be reduced to avoid the overhead of futile Caching.

For IMAGE integrity-sensitive applications where recovery is difficult or impossible, John and Al built a CACHECONTROL BLOCKONWRITE facility. This enables a user to force a process wait until the completion of physically posting the cache buffers to disc. This works at the systemwide level.

Alternatively, if IMAGE Logging and/or ILR (Intrinsic Level Recovery) are enabled by the database manager, the Serial Write Queue facility protects the integrity of the database.

A Performance Profile at Product Announcement

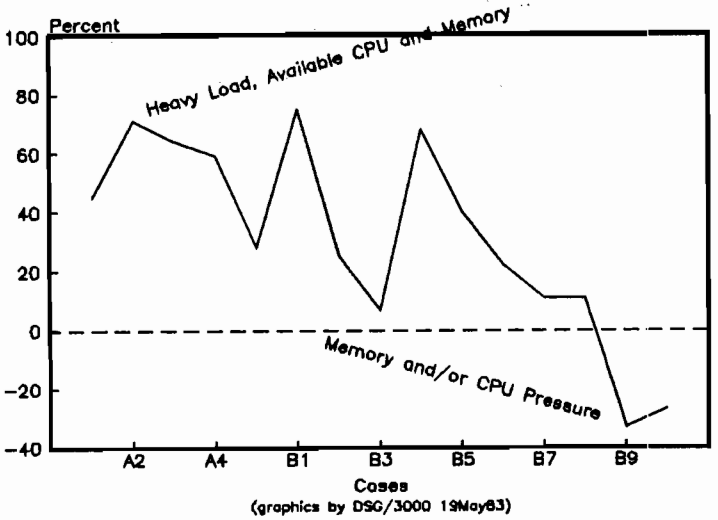
By the time Disc Caching was announced in May of 83, the Performance Center had benchmarked the product for four months. Just prior to the announcement a preview edition of the Performance News Notes (the monthly publication of the Performance Center) went to the field SE Organization with performance data like the Improvement Rate chart below.

This issue also had an internals description by John Busch and Al Kondoff. The combination of theoretical insight coupled with real-world performance data over a wide range of conditions gave the field SE's a head start in learning the characteristics of an important new product and the conditions that make it a good investment for the HP3000 user community.

Disc Cache: Improvement Rates

Groups A&B

Percent Change



About the Authors....

Becky McBride *is a Performance Specialist in the Systems Performance Center at the Hewlett-Packard computer facility in Cupertino, California. Becky's responsibilities include performance benchmarking and product characterization, with special emphasis on HP3000 application products. Becky joined HP in 1980, after receiving degrees in Computer Science and Accounting from Ohio State University.*

Sam Boles *is a Systems Engineer in the Systems Performance Center at the Hewlett-Packard computer facility in Cupertino, California. With HP for seven years, Sam's computer experience started back in the AUTOCODER days of the 1401/1410, migrated thru the 360/370 era, and now focuses on HP hardware and software performance characterization. Sam earned his MS at UCLA in Information Systems.*

sebiug47 0215 15Nov83

IMAGE: An Empirical Study

B. David Cathell
Hewlett-Packard

Background

Last April, Jorge Guerrero addressed the Performance Specialist Class that I was attending. He made a rather bold assertion that "There is no evidence that a prime capacity for an IMAGE master data set gives a better distribution than a non-prime capacity." In light of the fact that we had all been trained to believe that a prime capacity is decidedly better, we students were shocked to hear Jorge's contention. I began design and implemenation of a program to read any data base and test the key values contained in a specified data set using various capacities. Although the initial goal was to prove or disprove Jorge's assertion, the program was also intended to be a tool to evaluate potential capacities should the assertion prove to be true.

Image Theory

The following section deals with the theory upon which the IMAGE design is based. Readers who are already familiar with this theory may wish to jump forward to the discussion of the program itself.

IMAGE, like any data base system is an organized collection of data. The method of organization is intended to speed the retrieval of particular data that a user requires. There are many methods of organization but the designers of IMAGE chose two types of sets of data which we call master data sets and detail data sets.

Master data sets contain key data by which the rest of the data is retrieved. Names and identifying numbers (such as social security number) are common keys with which we are probably all familiar.

Detail data sets usually contain the bulk data but are organized with pointers to maintain linked lists of information with common key (called search items) values. Data for a particular person, for example, can be obtained by finding the head of the chain for that person's name and following the chain.

Because accessing the data in an IMAGE data base usually requires initially accessing a master data set, this paper will only deal with questions pertaining to masters.

When a user attempts to access a master data set, he or she provides a key value. Somehow, IMAGE must be able to transform that key value into a unique address where the desired record (data entry) resides. The easiest way to do this is to assign a unique number to every possible key value and have a correspondingly numbered data entry location reserved for it. Unfortunately, even a short data item may have a huge number of possible values; for example, the number of possible file names in MPE is over two trillion. Although, your HP sales representative might like to sell you another disc drive, two trillion records would require a good sized building full of disc drives.

Therefore, IMAGE has to transform that key value to some reasonable number of possible record locations. Whatever method is used must be repeatable. That is, once the user places information in some location, a subsequent request for that same information must be directed to the same location. Secondly, the method must produce a reasonably good distribution of record numbers. If the transformation algorithm produces the same record number from two different keys (the keys are called synonyms), we have a problem in that only one data entry can occupy a record location, but two entries wish to reside there.

Thus, in addition to selecting a good transformation algorithm, the designers of IMAGE also had to contend with a method of handling synonyms. The method chosen allows a synonym to reside in a location other than the one determined by the transformation algorithm. Such a synonym is designated a secondary entry, whereas an entry which resides in the correct location is termed the primary entry. The primary and secondary entries

which are synonyms of each other are linked together in a linked list, with the head of the chain in the primary location. In order to find a secondary entry, IMAGE must first find the primary entry and follow the synonym chain until it finds the desired entry.

Importance

With this brief discussion of the theory of IMAGE, one now has to ask the question "Why would anyone care if the capacity of a master data set is prime or not?" Actually, one has to go back to the transformation algorithm mentioned above.

IMAGE actually has two parts to the transformation. The first part applies only to byte type keys. It is called the Bale-Estes-White hashing algorithm and attempts to fold the entire key into a two word value. (The exact algorithm is given below.) The second part applies to all keys and consists of simple modulus arithmetic whereby the right most two words of the key (or two word hash) are divided by the capacity. The remainder plus one is the record location to be used for that key. It has been asserted (although I don't know who made the original assertion) that a prime divisor produces a better distribution of record numbers.

It should be apparent that a poor distribution would result in an excessive number of synonyms and thus a greater number of secondary entries. This results in additional overhead in accessing the master data set because IMAGE must more frequently follow a synonym chain in order to produce the desired data entry.

It can also cause additional overhead in adding or deleting entries in the master data set. This condition is termed migrating secondaries and results from a rule that IMAGE enforces; a data entry has first priority over its home (calculated) location. If in adding a new data entry, IMAGE discovers that a secondary entry (a synonym of some other key) already resides in the this data entry's home location, the secondary entry must be moved (migrated) first to some other empty location. Then the new primary key may be placed in its rightful home location.

The second case of migrating secondary is caused by the deletion of the primary entry which has a synonym chain. The second synonym in the chain is moved (migrated) to the primary location.

It should be obvious that it is very desirable to minimize the number of synonyms in order to avoid excessive overhead in both accessing and changing a master data set. If choosing a capacity which is a prime number (or any other specially computed number) produces significantly fewer synonyms, then it is certainly worth the effort.

The DBCAPCK program

In an effort to answer the question of prime capacities, I decided that one could argue the theory of transformation algorithms forever, but the proof is contained in real data bases with real user data. Although my primary objective was to prove or disprove the prime capacity assertion, I also intended that the design of my program would be flexible enough so that if the assertion proved to be true, the program could be used to determine good capacities for a given data set.

The first step was to obtain the IMAGE transformation algorithm and adapt it to my program. I had already decided to use PASCAL for the program and had to either interface to the actual SPL code or simulate it. Perhaps it was intellectual curiousity or perhaps some personality flaw, but I decided to simulate. For those who wish to share this experience, the hashing algorithm follows:

IMAGE Hashing Algorithm

1. Obtain the left most two words (4 bytes) of the key.
2. If the key length (in words) is an odd number, shift the two word value to the right by 16 bits, introducing leading zeros.
3. Shift the two words left by 1 bit, with an end around carry of the sign bit.
4. If all the words in the key have been used in the hashing, go to step 9.
5. Get the next two words in the key (beginning at the right) and divide (unsigned) by 31 and add 1 to the remainder.
6. Use the above result as a shift count and shift the hash accumulated thus far, to the left with an end around carry of the sign bit.
7. Perform an unsigned addition of these next two words in the key with the just shifted accumulated hash (saved as the new intermediate hash value.
8. Go back to step 4.
9. Shift the accumulated hash to the right by 1 bit, introducing a leading zero bit.

I can assure you that I had no confidence that I accurately translated the algorithm (written in SPL but actually every line is part of an ASSEMBLE statement). Therefore, the

program contains a check, comparing the record number computed by the program versus the actual record number of the primary key (aha! Finally a use for the mode 8 DBGET) when the user scans a data set specifying the original capacity.

Modulus Calculation

The actual calculation of the relative record number is very straight forward; one only needs to perform unsigned division of the hash (byte key) or right most two words by the capacity, using the remainder plus one as the relative record number. Or so I thought!

After the program was beginning to run, I visited one of my accounts who is a heavy data base user. They kindly consented to test out my program on one of their data bases. All was fine until we examined a data set that had a capacity of 150,000 entries (since it was non-prime, it looked like a good candidate to analyze). The program began issuing an error message indicating that the relative record number computed by the program differed from the data base. There were so many miscalculations that it couldn't have been an end case; in fact, we had the impression that every entry was miscalculated. But a pattern emerged, every calculated value was exactly one less than the value IMAGE had generated. Later testing revealed that IMAGE has a bug in the modulus calulation (I even suspected PASCAL) when the capacity is greater than 65535. Of course, the "bug" can never be fixed because of the requirement for repeatibility of the transformation algorithm.

General program description

When the program is run, it initially prompts the user for a data base name and password. It then opens the data base in mode 5 (shared, read only) and using DBINFO lists the master data sets with information about each. The user is prompted for a data set, a proposed capacity and a proposed blocking factor. The program reads the indicated data set, calculates the hash (if a byte key) for each data entry, stores it in an extra data segment, calculates the relative record number and records it in a statistics file. Once all the data entries have been read, it counts the number of keys which had no synonyms, the number with exactly one synonym and so on. It then produces a table of these results on the terminal display. The user may also request a graph of the distribution of entries. The user may then select to repeat with the same data set changing the proposed capacity. In the second pass of a data set, the program uses the hash from the extra data segment rather than re-reading the data set.

The following is a sample of the dialogue from the program.

```
DBCAPCK Version 1.0 (c) Hewlett-Packard Company, Inc.

                                Data Base = dummy
                                 Password = <cr>

No. Name            Type  BF      Entries    Capacity
 1  PEOPLE           M     14          166         200
 2  INT              M     68           36         200
 3  ASCINUM          M     38           25         200
                     Data set number = 1
                         Search item = NAME
                           Item type = X
                 Capacity (      200) = <cr>
               Next larger prime? (N) - <cr>
                 Blocking factor (14) = <cr>
                         Entry count = 166 (83.0%)
           Entries with  0 synonyms -          62    37.3%
           Entries with  1 synonyms -          46    27.7%
           Entries with  2 synonyms -          45    27.1%
           Entries with  3 synonyms -           8     4.8%
           Entries with  4 synonyms -           5     3.0%
               Overflow block count = 3
                  Total block count = 15
         Would you like a graph? (N) - <cr>
                         Repeat? (Y) - n
```

Most of the above is pretty straight forward, but a few items should be discussed. Why is blocking factor important? In all of the previous discussion, we simplified the explanation so that it would appear that each data entry has its own physical location on the disc. In fact, data entries reside in blocks whose blocksize is determined by the BLOCKMAX control option in DBSCHEMA. The number of data entries that will fit is a

function of the size of the data entries and the number of paths to associated detail data sets. As we discussed earlier, there is overhead associated with synonym chains, but the amount of overhead is much greater when the synonym chains span multiple blocks. The time required to perform the physical I/O to obtain a second (or even third) block in following a chain is considerably greater than if the chain is contained only within one block.

It is also for these reasons that the Overflow Block Count is reported. This statistic represents how many blocks in the data set could not contain all the data entries which have their home location in that block. In each of these blocks there will be at least one synonym chain which spans into another block. In many cases, this statistic may be as important as the actual synonym counts themselves.

Are prime capacities better?

I have used this program to examine many data bases with many types of keys and I cannot find any benefit to prime capacities. I realize that many of us find this hard to accept and may question the conclusion of one individual.

The following is a compilation of the statistics reported by DBCAPCK making numerous passes through a data set containing a six-byte key (TRACKER PICS id consisting of three character month abbreviation followed by three ASCII digits). The number of entries was 1243 with a blocking factor of 20. The results are representative of the patterns that I have observed.

	%	Percentage with synonym count of						Blocks	
Capacity	Full	0	1	2	3	4	5+	Ovrflo	Total
1380	90.1	40.9	36.8	15.9	5.1	1.2	0.0	19	69
1381*	90.0	43.4	34.6	15.9	4.8	0.8	0.5	20	70
1382	89.9	39.8	37.3	16.4	4.8	1.6	0.0	20	70
1383	89.9	40.5	38.9	15.9	4.2	0.4	0.0	18	70
1384	89.8	41.6	36.8	15.2	5.1	1.2	0.0	15	70
1385	89.7	41.4	37.8	13.3	6.8	0.8	0.0	15	70
1386	89.7	39.9	35.1	20.0	4.2	0.8	0.0	22	70
1387	89.6	40.2	35.4	18.8	3.5	2.0	0.0	18	70
1388	89.6	41.2	34.8	15.7	6.8	1.6	0.0	18	70
1389	89.5	41.5	36.5	15.7	3.9	2.4	0.0	16	70
1390	89.4	39.3	37.7	14.0	5.8	3.2	0.0	18	70
1391	89.4	40.1	37.7	16.4	5.5	0.4	0.0	19	70
1392	89.3	40.0	39.7	14.5	4.5	0.8	0.5	15	70
1393	89.2	40.6	34.6	17.1	6.4	1.2	0.0	20	70
1394	89.2	40.8	38.0	15.2	4.8	1.2	0.0	18	70
1395	89.1	43.3	37.0	14.7	4.2	0.8	0.0	16	70
1396	89.0	41.4	35.1	16.9	6.1	0.0	0.5	22	70
1397	89.0	41.8	36.8	13.8	5.5	1.6	0.5	18	70
1398	88.9	43.0	35.2	17.1	4.2	0.4	0.0	17	70
1399*	88.8	41.5	37.2	17.1	3.2	0.4	0.6	18	70
1400	88.8	43.5	36.5	13.3	5.8	0.4	0.5	19	70
1550	80.2	44.3	38.3	12.8	4.2	0.4	0.0	7	78
1551	80.1	45.7	38.0	11.3	4.5	0.0	0.5	10	78
1552	80.1	44.8	35.9	13.5	5.8	0.0	0.0	10	78
1553*	80.0	45.2	37.2	12.1	5.1	0.4	0.0	12	78
1554	80.0	46.3	36.0	12.6	3.5	1.6	0.0	10	78
1555	79.9	44.0	36.5	13.5	3.9	0.5	0.0	10	78
1775	70.0	50.1	36.7	10.6	2.6	0.0	0.0	6	89
1776	70.0	49.2	36.7	11.3	1.9	0.8	0.0	5	89
1777*	69.9	51.7	31.7	12.6	3.2	0.8	0.0	5	89
1778	69.9	51.6	36.5	9.7	2.3	0.0	0.0	6	89
1779	69.9	49.1	37.3	11.3	2.3	0.0	0.0	1	89

Prime capacities are marked with an asterisk (*).

General Conclusions

Are there especially good capacities?
Of the data sets that I have examined, there have not been any instances of a capacity that was overwhelmingly better than others of approximately the same value.

Are there especially bad capacities?
I have seen only one class of values that are clearly to be avoided; that is the powers of two. These capacities produce large numbers of synonyms, yet values one greater or less are perfectly acceptable.

What about percent full?
There does not appear to be a hard rule that some percentage is acceptable and one percent greater is unacceptable. Generally, a capacity between 70% and 80% seems to be satisfactory, especially in the overflow block count statistic.

So do I just pick a number out of the air?
I would suggest that you use a capacity that will keep your data set between 70% and 80% full and then if possible check it with DBCAPCK. If you feel comfortable with a prime capacity (or if you are forced to use a prime by third party software which will allow you to change the capacity of a data set), by all means, use it. Prime capacities don't appear to be bad, they just aren't demonstrably better than non-prime.

How do I get to run DBCAPCK?
I have sent a copy of DBCAPCK to the performance library at CSY in Cupertino. Contact the Performance trained SE in your area to obtain use of the program. Please do not try to contact the factory directly.

Conclusion

I think the real lesson to be learned by all this, is that as users we should all maintain a certain amount of skepticism when we are asked to accept "truths" which are not grounded in empirical evidence. And if you have your doubts, construct an experiment to test the assertion. You might overturn the next IMAGE myth. Why you might even discover that integer keys are all right after all!

B. David Cathell Born 1946 in Showell, Maryland. Raised in Baltimore, Maryland. Graduated from Fort Lauderdale (Florida) High School in 1964. Graduated from Purdue University in 1968 with a Bachelor of Science Degree in Mathematics with specialization in Computer Science, minor in Physics. Employed by Control Data Corporation in Arden Hills, Minnesota, as a Programmer Analyis. Designed and implemented Operating System software for the CDC 3000L Series Computer System. Transfered with Control Data to Sunnyvale, California in 1976. Designed and implemented Operating System software for the CDC STAR computer system. Employed by Hewlett-Packard Company in Cupertino, California beginning 1978. Designed and implemented Factory Automation software on the HP 1000 for internal use. Transfered to Neely Sales Organization in Santa Clara, California in 1980 as a Commercial (3000) Systems Engineer. Areas of specialization include Data Communications, Laser Printing, Performance, Personal Computing and Office Automation. Transfering to the Monterey Bay Sale Office when it opens, expected January, 1984.

Customer Satisfaction Through Quality Software

Dan Coats, Hewlett-Packard
Michael McCaffrey, Hewlett-Packard

Perhaps the most frustrating thing for a Hewlett Packard customer is updating to a new release of MPE. The majority of the older customers take the "wait and see attitude" before updating their production systems to the new software because they have been burned in the past. A few brave souls and new customer installations take on the task of being the guinea pigs. At a point, maybe 3 to 6 months after release, when the software has stabilized, the rest of the customer base is feeling more comfortable and gradually migrates onto the new software. This attitude, though justifiable, creates a number of problems for Computer Systems Division (CSY). Since the next release of MPE is based on the most current one in the field, we need a high percentage of customers using the current release to insure that we are not carrying problems forward into the next release. When bugs are found and fixes are generated for the installed customers, they are then incorporated into the next version of the product. Because of the time lag between the release of new software and the installation of it on a high percentage of the installed systems, we are working with a unknown quality level of software. What can be done about this CATCH 22 situation? We believe the most important thing is to raise your confidence level in our software. The only way to do this is to provide the highest quality software possible with the current technology available. With the stage set at this point, this paper will discuss some of the things we are doing at CSY to address this goal. This paper will first define the Product Life Cycle, which is a mechanism for defining the process of software development. The quality checks and testing at each phase will be covered in addition to what we are doing to automate the system testing to uncover as many problems as possible before software is released to our customers.

Software Product Life Cycle

The software development process is controlled by a document called the Software Product Life Cycle. This document defines the objectives, results or output, and method of validation for each phase of the life cycle.

This discussion will deal with a generic life cycle. While not exactly like the life cycle used at CSY, it defines the mechanism that most software divisions model their Product Life Cycles after, including CSY. There are six major steps in the process of developing software.

- o Investigation The requirements for the product to be successful in the market are determined. The product objectives, in terms of functionality, usability, reliability, performance, and supportability are defined.
- o Design External interfaces, both user interfaces and product interfaces, are defined. Internal specifications, which state the algorithms that will be used, are developed.
- o Implementation Code, module test, and document the product components. Plans are made for field and user training.
- o Testing Insure that the product meets HP and user requirements. Integration of the software. System testing, ALPHA and BETA testing. Field review and supportability evaluation.
- o Release Package and transfer to manufacturing and the field.
- o Support Ongoing enhancement and product fixes.

Software Product Life Cycle Testing Activity

In the early steps of the Software Product Life Cycle a great deal of testing activity occurs. The following chart shows the testing activity broken down for each step of the Product Life Cycle.

Phase	Product Development	Testing Activity
Investigation	Product Requirements	----------
Design	External design, and Internal designs	Design Inspections Quality Plans Test Plans
Implementation	Coding	Code Inspections Module Testing
Testing	Integration of the software	Function Testing System Testing ALPHA & BETA Sites Field Review

Quality Plans

The quality plan is a document that states the quality objectives of the project and the means for achieving them. These measurable objectives and specific plans will come from all functional areas and will result in goals agreed upon by the entire product team. As the project progresses, the Quality plan is used for evaluation and planning throughout the life of the product.

The quality plan is not a test plan, nor does the test plan deal with the functions of a quality plan. The scope of the quality plan is much broader, its creation and acceptance must coincide with those of the External Specifications in order to insure that the various aspects of the product's quality are considered at appropriate points in the product's life. A test plan, however, defines specific testing activity, is much more detailed and is created later in the product's development.

In order to successfully define the quality objectives of a product, the plan addresses each phase of the product life cycle and includes the following:

-specific objectives to be met;

-the means of achieving these objectives;

-what measures will be used to determine the degree of quality achieved;

-criteria needed before project enters the next phase;

-what documentation will be kept for all the activities.

The goals should be expressed in terms of fitness for use. There are five major aspects of fitness for use and each should be addressed for each phase of the project. These are:

-functionality

-usability

-reliability

-performance

-supportability

Quality Plans

Contents

The following table identifies the major components of the Quality Plan:

I. Product Overview

A. Product Identification

B. Related Products and Projects

C. Quality Perspective of Product

II. Plans by Phase

A. Design

B. Implementation

C. User Acceptance Testing

D. Release to Manufacturing

E. Post-Release

Quality Plans

I. Product Overview

A. Product Identification

1. Name, mnemonic, and number

2. Product Abstract

3. Project Personnel

B. Related Products and Projects

A list of all products and projects that affect this product is provided in order that dependencies among products can be identified.

C. Quality Perspective of Product

A description of the product emphasis, its market, and its potential end use is provided. In addition, problems that crop up as a result of the related products and project investigations are recorded and contingency plans are developed as well as methods for monitoring the problem areas.

II. Plans by Phase

For each of the five development phases objectives, plans, and methods of verification should be described. The quality control procedures should reflect specific measures to be taken to insure a high quality result at the completion of each project milestone; the quality improvement section focuses on what items or processes will be improved as a means of improving the resultant product's overall quality; the measurement section focuses on how the project team will know if its goals have been met. This outline will look at the objectives only; plans and verification methods will be dealt with later.

A. Design

The design phase consists of external and internal subphases which may overlap, depending on the needs of the project. In the external subphase the features of the product and the user interface are specified. The internal subphase defines the structure that will support the product's externals. During this phase, a test plan is also developed by the project team. This plan will be described later.

1. Objectives

External....

--to describe the product operating environment

--to determine the functional capabilities

--to define the user interface

--to specify documentation standards for the ES

Internal....

--to describe major elements of the internal structure

--to develop complete, detailed descriptions of the algorithms and data structures to be used in the implementationplementation

--to provide design alternatives

B. Implementation

This phase includes the coding of all modules, the completion of the internal maintenance specification, testing of the product, and manual writing.

User reference manuals, programming guides, and other manuals are completed so that they will be available for both alpha and beta test sites. This is an area known for a large number of product errors, therefore, particular care should be taken to review the manuals to ensure they are accurate, complete, and useful.

1. Objectives

Coding.....

--to determine coding standards to be used

--to provide idea of internal documentation expected

--to decide if hooks should be included for testing

--to faithfully translate the algorithms and data structures developed in internal design into programs, procedures, and data structure definitions

--to provide complete and functionally tested code

Internal Maintenance Specification.....

--to produce concurrently with the code

--to provide a complete description of the internal design, structure, and flow of a product

--to detail the development of the product

--to provide the information necessary to support and enhance the product

--to provide sufficient information so that field training may be developed

--to provide for evolution of the document after the product is released

Testing Activities.....

--to develop tests as outlined in the test plan

--to document tests in test documentation section

--to choose ALPHA test sites

C. User Testing

This phase exposes the product to a controlled end user environment. In order to get to this phase, all lab testing is completed, and the product has completed the implementation phase.

1. Objective

--to evaluate the initial reliability of the product

--to determine readiness for BETA site testing

--to provide active test site management

--to test the product installation files

D. Release

During this phase, the code must be tested extensively to insure its reliability, the performance must be measured and tuned until the product is ready for customer use, and control of the product must be turned over to manufacturing.

1. Objective

--to provide for thorough product testing through the use of the BETA test site

--to stress the product with the intent of causing it to fail

E. Post Release

The life of a product after release involves changes, both fixes and enhancements, the necessity to interface with new products, and the requirement to minimize disruption of existing customers' activities.

1. Objective

--to demonstrate that new releases have improved quality as measured by agreed upon metrics

--to test and prove that the code of the new release is maintainable

--to insure that documentation of the new release meets quality objectives

--to insure that advantages of fixes or features in the new release outweigh the risk of potentially introducing new bugs

--to reduce the time that a critical or serious problem remains open

The test plan is a strategy for applied testing of a product. The objective of a test plan is to define the scope, resources, and timetable of product testing. All of these will vary depending on the type of product and the objectives documented in the quality plan. The quality

plan makes specific recommendations about the testing process- the test plan implements them.

The test plan is the driving force for the testing process and as such, should address the following:

--the types of tests to be used by the project team

--non-standard test specifics

--frequency of repetitive operations (e.g., walkthroughs, regression tests).

It should define a set of tests which will be sufficient to guarantee the quality of the finished product upon release.

The test plan is developed as a integral part of the design and implementation phases; it is directly applied to the external specification.

The following is an outline of a test plan that is presently the standard for CSY. The intent, when developing such a plan, is to use only those areas that are directly applicable to the product being tested and this paper will only deal with some of the aspects of such a plan.

Test Plans Contents

I. Product Identification

A. Project name, mnemonic, and project number

B. Project Abstract

C. Project Personnel

II. Development Quality Control Practices

A. Design Walkthroughs

B. Design Reviews

C. Code Inspections and Walkthroughs

D. Code Reviews

III. Implementation Testing

A. Module or Unit testing

B. Integration testing

C. Development test tools

IV. Product Testing

A. Function testing

B. System testing
1. Facility
2. Volume
3. Stress
4. Usability
5. Security
6. Performance
7. Storage requirements
8. Configuration
9. Compatibility/Conversion
10. Installability
11. Reliability
12. Recovery
13. Serviceability
14. Documentation
15. User procedures
16. Certification

C. Acceptance testing

D. Installation testing

E. Test automation

F. Equipment and configuration needed for testing

I. Product Identification

This section is self-explanatory.

II. Development Quality Control Practices

In order to promote quality in the design and implementaion phases, the test plan delineates the ground rules and frequencies of walkthroughs, inspections, and design and code reviews.

III. Implementation Testing

A. Module or Unit Testing

Attempts to find the discrepancies between the external description of the module and its logic as demonstrated in executing the code.

B. Integration Testing

This involves combining the next module to be tested with the set of previously tested modules before it is tested. This tests the interface between two modules and the logic of one of them when the other has been previously module tested. The order used in integrating modules can simplify the testing process and have other important consequences.

C. Development Test Tools

IV. Product Testing

A. Function Testing

Looks for the discrepancies between the external functions as described in the external specifications and what the product actually provides.

B. System Testing

The plan selects which of the types of tests should be run; defines the objective and what verification criteria will be applied for each of the tests.

C. Acceptance Testing

This is the process of comparing the product to its initial requirements and the current needs of its end users. ALPHA and BETA test sites are the prime example.

D. Installation Testing

When installing many software products, a variety of options must be selected by the user; files and libraries must be allocated and loaded, a valid hardware configuration must be present, and the product must be interconnected to other products. Installation testing is the way to weed out these anomalies. This testing also occurs at the ALPHA and BETA sites.

E. Test Automation

F. Equipment and configuration needed for testing

Inspections, both design and code, are formalized processes used to find defects. The main idea behind inspections is to reduce or eliminate bugs as early as possible in the development cycle, thus increasing programmer productivity and lowering the product life cycle costs. Additionally, inspections provide a technically correct base for the next phase of the development cycle and insure adherence to the product specifications. The inspection team consists of four to five people who are assigned the following roles:

Moderator This is the key person for a successful inspection. The moderator is responsible for managing the inspection team, scheduling a suitable meeting place, reporting the inspection results, and following up on any rework.

Designer The person responsible for producing the design or code.

Reader Any team member other than the moderator or designer. Responsible for reading the material, one line at a time. Also has the responsiblities of an inspector.

Inspector Responsible for finding defects in the material being inspected.

The material to be inspected is distributed at least one week before the inspection is scheduled. Along with the material is the responsibility assignment for each member of the team. The formal inspection begins with the reader reading the document aloud, one line at a time. As defects are found, the moderator classifies and records them. Examples of such classifications are design logic errors, coding errors, interface errors, etc. The moderator faces two problems when the defects are found; one is to keep the remarks depersonalized and the other is to avoid solving the problem during the inspection. Keeping the remarks depersonalized is not an easy task. It is difficult for people not to say "you didn't..." or "you should have..." when they are discussing something they think the designer should have done differently. Instead, they should say "the design is wrong..." or "the code is...". The other thing to keep in mind is that the inspection is held to find defects. The resolution of the defects should be handled outside of the formal inspection meeting. The moderator should follow up on all defects to ensure they are corrected. The entire inspection should last no longer than two hours. It has been found that after two hours, the error detection efficiency of most inspection teams begins to dwindle.

A module, used in the structured programming sense, is a procedure or closely related set of procedures. If you look at two different approaches to software integration, you begin to see why module testing can be very important.

"Big Bang" Approach

1. Design, code, and test each module by itself.

2. Throw all the modules into a large bag.

3. Shake the bag very hard.

4. Cross your fingers and hope that it all works.

Big Bang Approach makes it very difficult to track down a bug. Which module is causing the error?

Incremental Approach

1. Design, code, and test one module by itself.

2. Add another module.

3. Test and debug the combination.

4. Repeat steps 2 and 3.

Incremental approach allows the process of debugging to be more scientific. Using the

incremental approach requires the use of stubs, drivers or both, when testing of a module begins. The first module is tested using dummy external calls. When the next module is ready, the dummy external call is replaced by a call to the new module. The tests against each module are saved and rerun with the addition of each new module. The incremental approach has allowed us to find things like interface errors and variable initialization problems very quickly. This not only simplifies the debugging process but produces a higher quality product in the end. Automated System Testing On any new release of MPE, there are over 800 job streams, containing over 10,000 separate tests, run against the operating system and the associated subsystems. In the past, operators worked around the clock, streaming each job and checking the output results for errors. Although this method did get all the tests run, one was never sure that all the errors were found and reported. Therefore, the decision was made to develop a series of tools to automate the testing process, in the hope of catching more errors earlier in the system testing process. The first program to be developed was the monitor process. This program is script driven and controls the streaming of all test jobs. We use user logging to record any errors in the testing process, and have written a set of intrinsics to allow the tests to report the P-relative code location of an error. In addition, system traps are armed by the monitor process to catch any abnormal aborts of the system intrinsics. With the controlling process written, the next step was to write a program to save the output spool results for later analysis. This program, called the ARCHIVER, is created by the MONITOR and copies all spool files to a tape using SPOOK. With the MONITOR and ARCHIVER in place, we began to catch a greater number of the test errors earlier in the testing process. The next problem was who was going to read thru 80,000 lines of spool output looking for inconsistencies. To solve this, a spoolfile comparision program was written. We took a known good run of the tests and checksummed them into a MPE file. The ARCHIVER now creates the spoolfile comparision program and passes each completed spoolfile to it for analysis. The spoolfile is checksummed and compared against the MPE file created from the known good run of the tests and errors such as missing lines, additional lines, and lines that do not match are reported. The automated testing has allowed us to catch a large number of defects in the software that would otherwise have reached the customer. Projects are now in process to enhance the automated testing for more functionality and to increase the testing coverage of the operating system.

Bibliography

Software Product Lifecycle. Hewlett-Packard; SPLC.83.02 #5955-1756; February, 1983.

Software Quality. DAS Quality Management Team, Information Networks Division, Hewlett-Packard; July, 1981.

A Guide to Writing Quality Management Plans. SNA Quality Management Team, Information Networks Division, Hewlett-Packard; August, 1981.

Quality Plan. Memo from Sally Dudley, Hewlett-Packard; May, 1980.

An Overview of Hewlett-Packard's Software Life Cycle Preliminary from Sally Dudley, Hewlett-Packard; September, 1983.

Design Inspection Seminar Jim Dobbins, IBM; January, 1983.

QUALITY AND PRODUCTIVITY IMPROVEMENTS IN THE PRINTED CIRCUIT ASSEMBLY PROCESS THROUGH THE USE OF STATISTICAL QUALITY CONTROL

F. TIMOTHY FULLER

THE HEWLETT PACKARD COMPANY
SANTA CLARA, CALIFORNIA
SEPTEMBER, 1983

ACKNOWLEDGEMENTS

The establishment of a successful SQC (statistical quality control) program at the Computer Systems Division of Hewlett Packard Company has been achieved only through the dedicated efforts of many people; I would like to give special recognition to several:

To Mike Forster, Manufacturing Manager, and Ilene Birkwood, Q.A. Manager, who had the foresight to see the benefits of this program and who provided the encouragement and the resources to get us moving.

To Dr. Perry Gluckman for bringing us the knowledge of SQC and showing us how to make it work and also for providing the motivation necessary to keep me hacking away at my HP120 long enough to produce this report.

To John Gilson and his team of wave solder operators who experimented with these concepts and gave us the first major breakthrough in process improvement.

To Mike Martinez and his Automatic Insertion team who developed SQC tools described in this paper to dramatically improve the AI process.

To Bob Tellez who led the material discrepancy project, who provided encouragement to his entire PC Assembly department to implement SQC, and who has learned how to change the process to make improvements.

To Jack Cambra who summarized mountains of data to help improve the board testing process and to help monitor Bob's assembly process.

To Dr. Spencer Graves who volunteered his time to work with our study group, who watched the progress of our program and developed the material in Appendix A, and who read and re-read this paper and provided many ideas to improve it.

To Vince Roland, Tim Pierce, and Tim Vachon who provided many helpful comments to improve this report.

And to all the other members of the Pre-Fab Department who worked so hard in our study group, collected data, and worked on the many successful projects not mentioned here; to Tim Metcalf's Q.A. group who worked with us to improve our inspection procedures, helped train our production workers, and designed and made our visual aids; and to all the other managers and staff who participated on improvement teams, worked on projects in their departments, and put up with my continual raving about the benefits of SQC for so many months.

CONTENTS

INTRODUCTION

In July, 1982, the Computer Systems Division (CSY) of Hewlett Packard Company began the task of implementing Statistical Quality Control (SQC) in its Manufacturing area. After 13 months of concentrated efforts a number of successful projects have been completed and dramatic gains have been made both in quality and in productivity. This paper will attempt to trace the events leading up to the SQC program, provide a detailed description of the several completed projects, and describe the conditions necessary for successful implementation of a similar program at the reader's own division or company.

In addition a model of SQC project process flow is presented in Appendix A. This model was constructed after observing several successful projects. The detailed description of each step should help guide the new SQC practitioner through the sometimes difficult path that will lead to substantial improvements in quality and productivity.

A brief definition of SQC as used at CSY will help guide the reader through the rest of this paper. The purpose of SQC is to provide a framework to allow an organization to work toward continuous improvements in product and process which will increase customer satisfaction at minimum cost. The tools, which are not described here in detail, include control charts, process flow diagrams, cause-effect diagrams, Pareto charts, and others; the bibliography provides sufficient sources for the reader to

acquire the necessary details. The process, which is described in Appendix A, involves the selection of a condition which needs to be changed, collecting and analyzing data, and deciding when a process should be changed to improve results. This paper will not explicitly define the types of changes needed; these are necessarily left to management who, through competent use of these techniques, gains a thorough understanding of their own processes.

HOW CSY GOT STARTED WITH SQC

In March, 1981, Dr. W. Edwards Deming was invited by consulting statistician Dr. Perry Gluckman, on behalf of top HP management, to give a two-day seminar to high-level managers from numerous divisions of the company. The seminar stimulated thinking about SQC in a number of HP divisions, especially CSY Manufacturing. Soon after, Dr. Gluckman was hired to teach an introductory course in SQC to managers in the CSY manufacturing area. He was selected because of his strong background in statistics, his knowledge of Deming's teachings and his prior consulting work at HP and other companies. The 16-hour course covered basic statistics, the red bead probability demonstration, and the use of control charts. Although interest in SQC was high for a short time during and after the course, no significant projects were carried out.

In June, 1982, Gluckman was again hired by CSY manufacturing, this time for a longer term. The objective was to stimulate the workforce to undertake some SQC projects to improve

quality. He was to be available one day each week to consult with anyone who felt motivated to begin collecting data and to start a project. He also met informally with various departments to suggest areas to be studied. In the beginning no formal objectives were set for SQC implementation. However, there was a general feeling that it was time to get started, attempt some projects, and see what happened.

ORGANIZATION AND DESCRIPTION OF CSY MANUFACTURING

The mission of CSY Manufacturing is to assemble and test HP3000 general purpose business computers. It is organized into departments as follows:

PC ASSEMBLY. Blank printed circuit (PC) boards and components are assembled using automated and hand loading of components, wave and hand soldering, and various other operations including masking, forming, and assembly of fabricated parts.

PC TEST. Completed PC assemblies are turned on and tested using various types of automated and manual test equipment. Failing and misloaded components, solder defects, and internal board problems are discovered and repaired in this area.

CABLE ASSEMBLY AND TEST. Cables and harnesses, which connect printed circuit boards together inside the system, are assembled using both manual and semi-automated techniques and equipment. Assemblies are tested for continuity and repaired if necessary.

FINAL ASSEMBLY AND TEST. Sub-assemblies and fabricated parts are assembled together to make completed systems

which undergo various levels of automated testing. Some repairs are made to defective sub-assemblies in this department; defective sub-assemblies may also be returned to previous manufacturing areas to be reworked.

SUPPORT DEPARTMENTS. Departments which provide support to the production areas include Order Processing, Production Control, Purchasing, Incoming Inspection, Stores, Information Systems, Process Engineering and Production Engineering.

THE WAVE SOLDER PROJECT

Project Set-up.

The first project undertaken was to attempt to measure the quality of solder joints which were being produced by the wave solder process. Problems had been evident for some time as many defective solder joints were being detected in later stages of the manufacturing process. This was occurring even though all soldered boards were inspected and touched up immediately after the wave solder process.

The first step was to present a training class to the operators on the wave solder team. They were given instruction on how to collect data and plot points on X-bar and R control charts. During the training it was emphasized that this was to be a team problem solving effort and that management was firmly comitted to the belief that most defects were caused by problems inherent in the process itself and were not the fault of the operators. It was also emphasized that the operators' responsi-

bility was to help identify problems and that corrective action would be the job of management.

The first task was to decide what would be classified as defects, how these defects would be measured and recorded, and how the data would be plotted for later analysis. The wave solder team, with some help from the QA department, made a list of the defects which had been encountered in the past and made up a check sheet which would allow the data to be recorded. A sample of the check sheet is shown in Figure 1. The check sheet listed the various types of defects that might be found on both the component side and circuit side of the board and spaces were provided for writing in the number of each type of observed defect. A grid was provided so that the location on the board of each defect could also be recorded.

It was decided that a 10% sample of soldered boards would be inspected and data recorded on the check sheets. As each board exited the wave solder process, the operator was instructed to roll a multi-sided die. If the roll was a "7" (probability of 0.1), the board was delivered to an inspector who was instructed to find, classify, and record the defects. The inspectors had received extensive training from the QA department so they could properly identify defective joints.

After recording data for a few days it became clear that the process was severely out of control and that approximately 1.8% (18,000 PPM) of the solder joints were defective.[1] A process engineer was then called in to study the current operating procedures. A matrix was developed which showed the values of

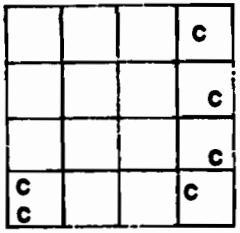
WAVE SOLDER DEFECT CHECK SHEET

INSPECTOR: 27
DATE : 8-25
TIME : 9 A.M.

ASSEMBLY: 47-14
LOT NO. : 6605
ORDER NO: 3215

COMPONENT SIDE

A) SOLDER SPLASH	____	
B) RAISED COMPONENTS	____	
C) INSUFFICIENT SOLDER	6	
D) SOLDER BRIDGES	____	
TOTAL		6



CIRCUIT SIDE

A) COMPONENT LEAD	____	
B) COLD JOINT	1	
C) NONWETTING	____	
D) SOLDER BALLS	____	
E) BLOW HOLES	5	
F) ICICLES	____	
G) SOLDER BRIDGES	____	
TOTAL		6
TOTAL BOTH SIDES		12

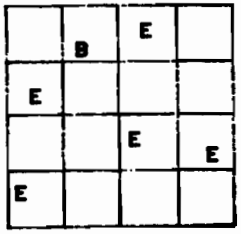


FIGURE 1. Wave Solder Defect Check Sheet. This shows an example of a completed check sheet. Note that 3 types of defects were observed and that "Insufficient Solder" defects tend to cluster along the edges of the assembly while "Blow Holes" appear to be distributed randomly.

the critical variables such as conveyer speed and solder temperature which could be adjusted for each type of assembly. New operating procedures were then written and displayed. An immediate reduction in defects was noted as the new procedures were put into effect by the operators.

Pareto analysis of the defects then showed that a large number were blow holes and that high defect rates seemed to be associated with certain lots of raw boards. A raw board manufacturing problem was suspected so a meeting was called with the supplier to discuss the problem. The wave solder group decided to begin incoming solderability testing of a sample board from each lot; if the sample showed poor solderability, the supplier was instructed not to ship the lot until the problem could be found and corrected. This procedure eliminated blow holes as a major category of defects. Two months of work and some simple process changes had reduced the defect rate to approximately 2,000 PPM.

Further analysis of the data showed that insufficient solder in the holes was now the major cause of defects. It was observed that most of these defects were associated only with the IC pins attached to the ground plane inside the board; insufficient board temperature was the most likely cause so an experiment was performed. The experiment consisted of running the boards through the pre-heater twice before applying the solder in order to raise the temperature of the internal ground plane. The experiment successfully eliminated the problem; since it was determined that modifying the equipment to increase pre-heat would be difficult, procedures were changed to run boards

through the pre-heater twice. Even though this procedure was quite out of the ordinary, the fact that it was developed by the operators themselves helped ensure that it was followed faithfully.

After collecting data on the revised process it was noted that in addition to eliminating the insufficient solder problem, other categories of defects seemed to be lower also. One suggested explanation of the improvement was that since flux was applied before the first pass through the pre-heater, the longer activation time improved the process. At this same time the operators observed that the conveyor occasionally halted briefly and that boards on the conveyor during these halts tended to have more defects. Maintenance was called in and the problem was corrected. Control charts now indicated that defects were less than 100 PPM.

With the significant reduction in defects it was now nesessary to modify the data collection procedure to accumulate more defects before data was plotted. The new procedure was to total the number of defects over 4 boards to make one data point. A question was now raised as to whether the inspectors were really counting all defects. A new procedure was established such that QA received a sample of the inspected boards, re-inspected, collected data, and compared their findings with data on the original check sheets. Their data indicated that the inspection process was accurate.

Push Toward Zero Defects.

As the wave solder team became more expert at collecting and

analyzing defect data, a feeling was spreading that achieving a defect-free solder process was possible. However, to accomplish defect-free operation, many small causes would have to be found and eliminated. During the next few months a number of small projects were carried out. Solder rack dimensions were checked and a number of fixtures were either repaired or discarded. Flux density was monitored and a new procedure was instituted to better control the density. Board surface temperature was monitored by attaching thermal probes to the top of the boards as they emerged from the pre-heater. A new tool was developed to check the evenness and width of the flux and solder waves. Data indicated that some IC sockets were not soldering properly so a different type socket was specified by engineering. It was determined that partially loaded assemblies that were sometimes placed on shelves for several weeks awaiting the arrival of backordered parts often had higher levels of defects. A process change was made to reduce lead time by delaying board loading until all components were on hand.

During the month of July, 1983, the defect rate was calculated to be 6 PPM on day shift and 15 PPM on swing shift. The wave solder team is continuing to collect and analyze data so information can be provided to determine how to further reduce the defect level. It appears that gains may be made by studying the automatic insertion cut-and-clinch process and incoming component solderability. Figure 2 shows a monthly summary of the dramatic reduction of solder defects in parts per million.

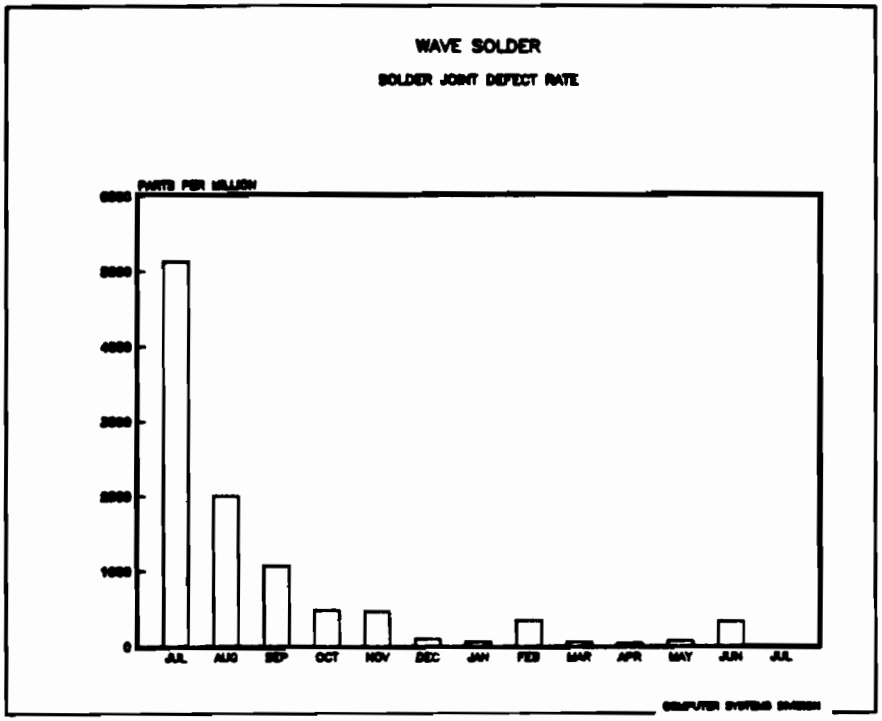


Figure 2. Wave Solder Defect Rate. This shows the dramatic decline of solder defects over a 13-month period. The large increase in June, 1983, was attributed to one lot of boards which had sat unattended on a shelf for several weeks.

Effects of Solder Defect Reduction on Other Operations in the Assembly Process.

A major impact of reduced solder defects was the elimination of Touch-Up as an operation in the assembly process. Previously, soldered boards were returned to an assembly team which was responsible for inspecting the assemblies for poor solder joints, solder balls, solder bridges and other defects. It was left up to each production worker to decide which joints were defective and to take the appropriate corrective action. This operation was considered very difficult and required the most experienced and skilled workers.

This operation itself caused numerous defects including board damage, burned traces, and heat-damaged components. Some of these defects produced scrap in the Touch-Up area while others caused failures and repairs in the Turn-On Test area. Still others undoubtably resulted in expensive warranty failures but no data is currently available to support this. Also, hand soldering left a rosin residue on the IC leads which had to be removed by a special cleaning process so good electrical contact would be made by bed-of-nails test fixtures in the Turn-On area.

After reduction of the defect rate and elimination of the touch-up operation, solder-related defects found in the Turn-On area declined from .05 defects per board to .01 defects per board. It should be noted that the number of defects discovered in the test area was less than those counted by the solder operators because the criteria used by the wave solder inspectors were very tight such that many of these defects were very minor and did not cause electrical failures. After the solder

defect level was reduced it was observed that production workers were still performing extensive touch-up to each board! It is probable that the workers felt obligated to find and fix defects, and since they were not well trained in understanding exactly which joints were defective, were repairing good joints. The process was changed to eliminate the Touch-Up operation and standard labor times for board assembly were reduced.

THE AUTOMATIC INSERTION PROJECT.

Process Description.

Another project involving SQC was to study the process of automatic insertion (AI) of dual in-line packaged (DIP) integrated circuits (IC's) onto boards. The equipment in use was an Amistar Model CI1000 automatic inserter which was capable of inserting .300 in. center DIPS. Kits of parts were delivered to the machine and tubes containing 25 IC's were loaded into the 64 stations of the machine. Approximately 10 to 50 boards were loaded in a batch; IC tubes were replaced in the stations as tubes were emptied.

After the components were loaded onto the boards an inspection was performed. A clear plastic sheet had been designed for each assembly; the sheet had the outline of each component and its part number printed on the surface. The inspection process consisted of placing the sheet over the assembled board and visually comparing the markings on each component with the information printed on the plastic sheet.

When mistakes were found, the inspector replaced the incorrect parts. When the project started, no data was being kept by the inspector.

Process Characteristics.

Before SQC techniques were applied to the AI process many operational problems were evident. The machine often malfunctioned and caused parts to be mis-inserted or damaged. The machine operator spent a significant amount of time inserting parts by hand, repairing damaged parts, and clearing jams from the machine. The equipment was often out of service while the maintenance department worked to correct problems. Breakdowns lasting several days were not uncommon while replacement parts were air-shipped from the manufacturer's factory. Often large queues of work-in-process (WIP) were formed and weekend overtime was required to catch up with the workload. The process had been operating as described for approximately two and one-half years.

Implementation of SQC Techniques.

In September, 1982, the supervisor of the AI department embarked on a project to introduce SQC techniques to the process in hopes of improving the situation. With help from Dr. Gluckman and information gained from the wave solder project, a check sheet was developed to provide a means for the operators to begin collecting data on the process. The check sheet listed 12 of the most common problems which had been experienced by the operators. For this project only machine-related problems were studied; problems such as "wrong part inserted" were left for

AUTO INSERTION
DATA COLLECTION SHEET

ASSEMBLY 01-14
QUANTITY 1026
DATE 9-30

PRE-INSERTION DEFECTS	QTY.	POST-INSERTION DEFECTS	QTY.
	10		

A. DIP IN MAGAZINE
B. DIP IN SPRING |||
C. STUCK IN SLIDE
D. ROTATOR ||||
E. FORMER
F. DIE RETRACT
G. STUCK IN JAWS
H. MISSING IC'S
I. DAMAGED/DEFORMED ||
J. BENT LEGS
K. WRONG VALUE
L. REVERSED POLARITY
M. TILTED IC'S

PERCENT OF DEFECTS

A.	%	E.	%	I. 20	%
B. 30	%	F.	%	J.	%
C.	%	G.	%	K.	%
D. 50	%	H.	%	L.	%
				M.	%

Figure 3. Automatic Insertion Check Sheet. Usually, the number of defects counted in a sample of four boards were tallied on each check sheet. In this example 10 defects were recorded out of 1026 components which were inserted.

later analysis. An example of the AI check sheet is shown in Figure 3.

The operators were given some basic instruction in SQC methods and data collection procedures were established. The operators were instructed to roll a multi-sided die immediately before beginning to insert a board. If the die showed a "7" (probability of .1), data on defects would be collected for that board. As each problem was observed by the operator it was tallied in the appropriate category on the check sheet. One check sheet was used to accumulate defect data for 5 assemblies. When the check sheet was complete, the total number of defects for 5 boards was determined and recorded on an X-Bar and R control chart. Data from the check sheets was also summarized using Pareto charts. A sample of the Pareto chart is shown in Figure 4.

After collecting data for a few weeks it was determined that defects were occurring at the rate of approximately 3% (30,000 PPM) and that there was significant variability in the process. Pareto analysis showed that two classes of defects, "DIP Stuck in Spring" and "DIP Stuck in Rotator" accounted for the majority of the defects. Based on this data, two changes to the process were made. First, maintenance was asked to add magazine cleaning and rotator adjustment to their regular semi-monthly preventive maintenance procedures. Second, springs were checked and worn or damaged springs were replaced. After more data was collected it was observed that these two classes of defects noted were significantly reduced.

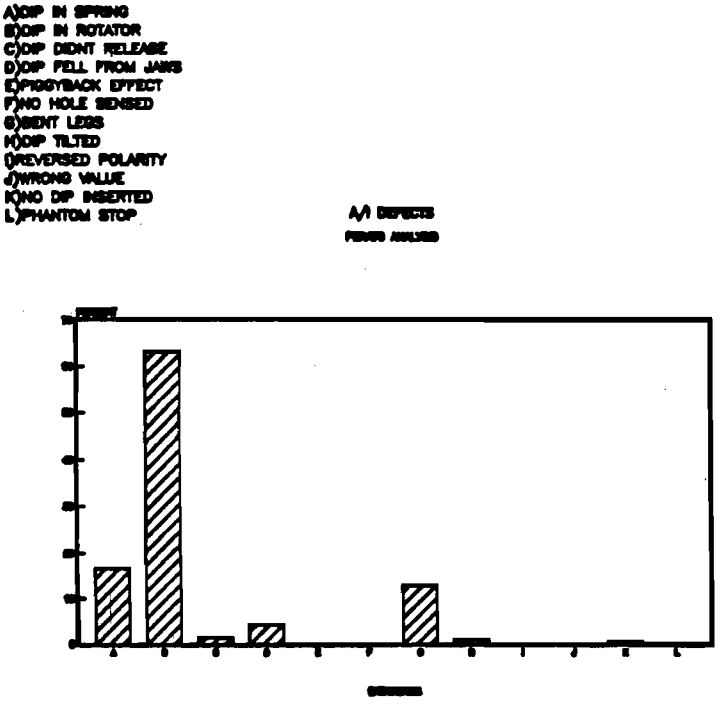


Figure 4. Pareto Analysis of Auto Insertion Defects. Note that in this example more than 60% of the defects wer "Dip Stuck in Rotator".

Data collected for the month of November, 1982, indicated that the major problem was "Phantom Stoppage" of the machine when it was about to insert a DIP. Neither the operators nor the maintenance department were able to determine the cause of this problem. Although greatly reduced, "DIP Stuck in Rotator", was also still a major cetegory of defects. At this point the AI supervisor requested that he be allowed to attend a training class given by the manufacturer of the equipment. During this class he showed the defect data which had been collected and worked with the factory to determine the major cause of "Phantom Stops". It was determined that poor design of the hole sensing unit in the machine was the cause of "Phantom Stops". He also learned that an improved rotator had been designed by the factory and was available to be ordered. Upon returning to the Division, the supervisor modified the sensor unit as instructed by the equipment vendor and the problem of "Phantom Stops" was eliminated. A new rotator assembly was also ordered. After several months of work the mean defect rate was now running at 1.5% (15,000 PPM). Up-time on the equipment was now better than 90%, up from less than 50% at the beginning of the project.

During the months of February and March, 1983, further improvements were made to the equipment based on continued analysis of the defect data. The new rotator assembly was received and installed and a worn-out head assembly was replaced. While watching the insertion operation the supervisor noticed that the boards tended to vibrate rapidly as the insertion head completed each insertion cycle. With assistance from the Tooling department, support tabs were added to the tooling plate to

improve board stability. The defect category "Bent IC Legs" was significantly reduced after the addition of the support tabs. During this time a concern was raised that the hole diameters on raw boards might be smaller than specified and could be causing mis-insertions. A study of hole diameters was carried out by the operators which showed the holes to be within specs and slightly oversize which ruled this out as a defect cause. Mis-insertions for March, 1983, were 1.1% (11,000 PPM).

During the next two months a number of adjustments were made to the equipment to further reduce the remaining defects. An improvement was made in the rotator-to-preformer alignment and the rotation chamber was readjusted to ensure an exact 90-degree turn to properly position the DIPS for insertion. One of the major categories of defects was now "DIP Didn't Release". Study of this problem indicated that a modification could be made to the magazine stations which would allow for better travel of DIPS through the stations. The modification was made to all 64 stations and this category of problems was reduced. These actions helped lower the defect rate to approximately .75% (7,500 PPM) for the month of May.

During the month of June it was noticed that a higher than normal defect rate was experienced when plastic and ceramic parts were mixed in the same lot. A combination of adjustments to the interposer and also to the height of the preform chute allowed the machine to insert both types of parts. A milestone was reached as the operators recorded 100% up-time of the machine for the month. It should also be noted that turn-around time for

batches of boards processed through the AI department was now less than 5 hours; this figure compares favorably to turn-around average of 20 hours earlier in the project. The defect rate recorded for June, 1983, was .56% (5,600 PPM).

Summary of the AI Project.

During the first nine months of the Auto Insertion project approximately 80% of mis-insertion defects were eliminated from the process. The changes made to reduce defects produced equally dramatic improvements in turn-around and equipment up-time. Also, the working environment was improved for both operators and maintenance people as the management emphasis was on the collection and study of data, not on the blaming of operators and technicians for mis-insertions.

The AI project has not ended; the goal for mis-insertions is zero. The AI team understands that further improvements will come only by their continued study of the process to help guide management in making necessary changes to the process.

THE MATERIAL DISCREPANCY PROJECT

The Discrepancy Problem.

One of the major problems faced by the production workers in the assembly area was finding an effective way of correcting problems in the kits of parts used to assemble printed circuit assemblies. Workers often had to spend a significant amount of time filling out requisitions for missing parts, returning excess parts to stock, and exchanging wrong parts before starting work on a batch of assemblies. It was felt that the application of SQC techniques to the process of assembling kits could help improve the situation.

The Process of Assembling Parts Kits.

The parts storage area had been recently relocated next to the assembly area in order to improve the flow and control of part issues. Daily, Stores received a batch of pull documents for each work order which was scheduled to be pulled the next work day. The batch contained one pull tag for each part type to be included in the work order kit. Kits contained enough parts to build from 5 to 400 printed circuit boards of a particular type; 10 to 20 kits were pulled and assembled on an average day and each kit contained from 10 to 100 different part types with the average being approximately 50.

The process used by Stores began by sorting the pull documents into part number sequence so that multiple quantities of common parts could be pulled at the same time. When pulling was complete, parts were placed in large containers labled with

the proper workorder number. Large and expensive parts were counted while small, inexpensive parts were weighed to determine proper quantities. When a workorder kit was complete, an audit was conducted to determine if all the necessary parts were present in the proper quantities. Data was collected during the audit and the indicated error rate for pulling was less than 1%. Completed kits were then delivered to the assembly area.

Data Collection and Analysis.

The first step in the project was to begin collecting data on discrepancy problems found in the assembly area. It was decided that a check sheet would be filled out for each kit as the assembly process was begun. All errors were recorded on the check sheets and an X-Bar and R control chart was begun. Check sheets were accumulated so a summary of error types could be prepared and analyzed.

After the data collection procedures had been in effect for two weeks the control chart indicated that the pulling process was in control and showed an average error count of 1.3 errors per kit with a range of zero to 8 errors. The data indicated that the audit procedure in the Stores area was not effective in screening errors from the assembly area.[2] The data was shown to the Stores group and, after some convincing, they agreed to begin using the new measure, "discrepancies per kit", to determine the quality of the pulling effort. [3]

Since the solution to the problem would involve close cooperation between two departments, a Quality Action Team (QAT) was formed with supervisors from the Assembly and Stores areas.

Discrepancy data was then analyzed by the team and it was observed that the errors could be categorized as "wrong part received", "mixed parts received", "parts not received", "overage received", and "shortage received". The team then brainstormed a list of possible causes for discrepancies; a cause-and-effect diagram showing possible causes is shown in Figure 5.[4]

Various members of the team were then assigned to analyze various parts of the pulling operation and to make recommendations for improvements to reduce errors. One group looked at the weighing process and found that the scale, although very accurate, produced shortages and overages of many parts due to the variation in weights of individual parts in a lot; weighing was also influenced by air currents from the air conditioning system. Several actions were taken to solve these problems including ordering a plexiglass shield for the scale, implementing use of an unused counter-bagger for certain parts and creating a bin stock of frequently-used, low-value parts in the assembly area. The bin stock allowed shortages and overages to be easily corrected in the assembly area; shortages and overages of these low-value parts were no longer counted as errors.

To reduce human error in pulling, part-identification training was provided to stores personnel and sample parts were attached to each bin to help reduce stocking errors. An inexpensive resistor-counter was ordered to improve the accuracy of resistor counts. Data was collected on the accuracy of counts of parts which were pre-bagged by vendors; significant variations were found among vendors including several which consistently

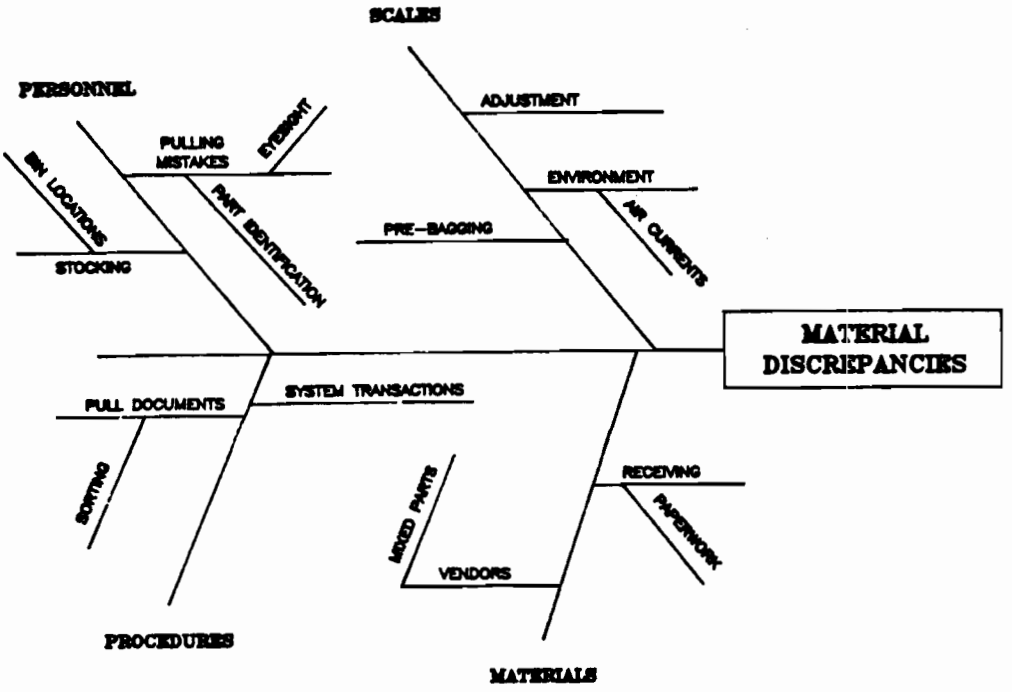


Figure 5. Cause and Effect Diagram Showing Possible Causes of Material Discrepancies. Brainstorming is a good technique for identifying possible causes for a problem such as this.

shipped less than the quantity labeled on the bag. Purchasing was called in to help correct the problem.

Results of the Project.

As improvements were made to the pulling process, the error rate showed a dramatic reduction; after two months the discrepancy rate had dropped from 1.3 errors per kit to less than 0.2 errors per kit. Productivity in the assembly area improved as less time was spent correcting discrepancies. A major improvement was also made in the process of sending kits to a local board assembly subcontractor. Previously, the subcontractor had been allocated 180 square feet near the Stores area to audit kits before they were sent out for assembly; a full-time inspector performed inspection and resolved discrepancies. After being shown the data indicating the improvements in pulling accuracy, the subcontractor agreed to eliminate his inspection procedure. This action improved the subcontracting process and opened floor space for other uses.

Two important conclusions can be drawn from the results of this project. First, significant process improvements can be realized in a short period of time through the use of simple, accurate methods of collecting and analyzing data. Second, a team approach allows adjacent departments to work together to understand each other's processes and make improvements.

CYCLE TIME IMPROVEMENT PROJECT

Printed Circuit Assembly Process.

As favorable progress continued in the areas of reducing defects and errors, an improvement in the work flow was becoming apparent. It was decided that good results might be obtained by studying the overall assembly process to see if the manufacturing cycle time could be reduced; cycle time was defined as the total calendar time elapsed from the time kits of parts were delivered to the assembly area to the time the last tested board in the lot was shipped to the final assembly area. [5] Each kit contained enough materials to assemble approximately one week's usage of each particular type of assembly. Approximately 50 different assemblies were built and quantities varied from as few as 5 for infrequently needed boards to 400 for some high volume types.

The Improvement Process.

The first step in the improvement process was to analyze data that was already being collected to determine how long it was currently taking to complete lots of boards; the standard lead times used in the MRP (computerized material planning and control) system varied from 11 to 15 work days for each board type. Although weekly quantities were used as workorder sizes for all assemblies, lead times were varied by up to 4 days in an attempt to provide some smoothing of the workload. The data indicated that for assemblies which followed the normal flow through the process, the average cycle time was 16 work days with a range of 9 to 36 days. For assemblies that were loaded by

subcontractors the average cycle time was 25 work days with a range of 10 to 45 days. The extra time needed for subcontracted assemblies was assumed to be caused by the extra handling and transit time required and the difficulties associated with replacing lost or damaged parts.

The next step was to construct a detailed process flow diagram to show each operation in the assembly and test process. Figure 6 shows the diagram which was constructed for one type of PC board. A circle was included for each separate operation which was performed; a triangle was shown for each shelf or staging area between operations. [6] As there were several possible sequences of operations depending on board type, the assembly with the highest material content was diagrammed first; since this assembly required IC's to be assembled into sockets, and the automatic insertion equipment lacked the capability to insert sockets, the AI operation is not shown for this assembly. It should be noted here that although the entire process required build times of several weeks, only a few hours of direct labor or machine time were actually recorded.

Copies were then made of the process flow diagram so they could be used as data recording sheets. Twice each day, at 9 a.m. and 1 p.m., a supervisor walked through the assembly area and recorded the location in the process of each board on the process flow diagram. Since the cycle time was more than three weeks and 2 to 4 lots of boards were in various stages of assembly at the same time, lot number information was also recorded.

After several weeks of data collection the data was summarized using a "Spencer Diagram" which graphically showed

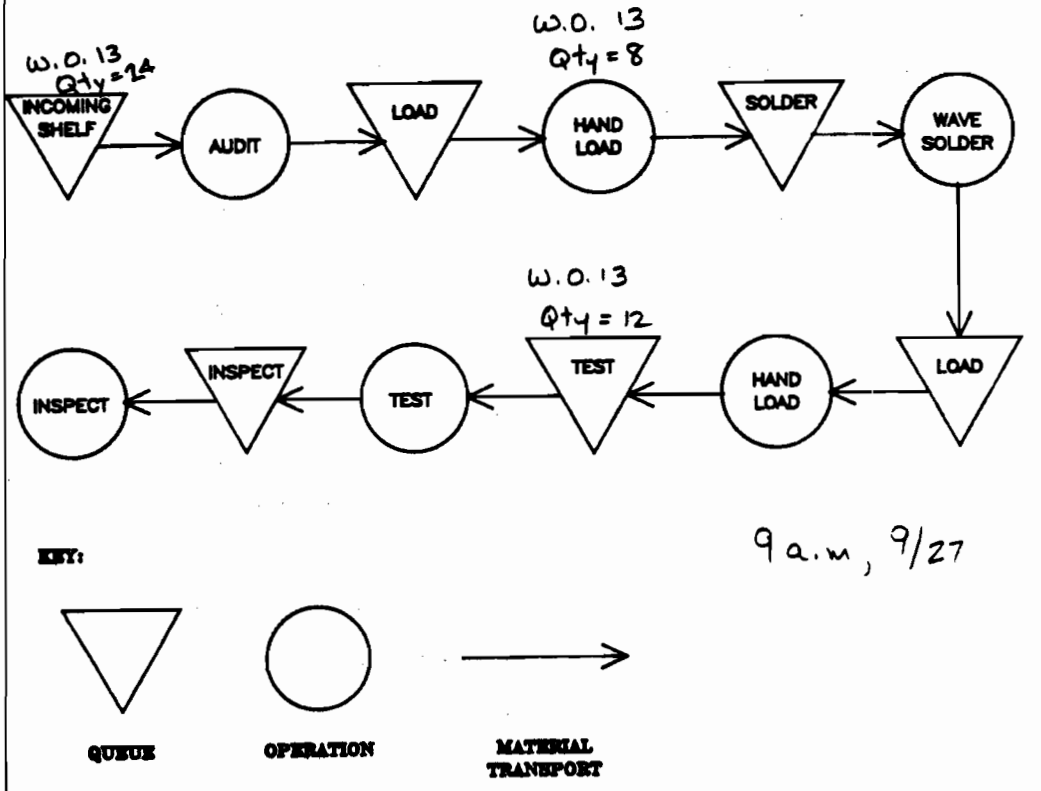


Figure 6. Process Flow Diagram. This example shows the process flow diagram used for data collection. The work order number and quantity of boards found in each step of the process has been recorded. In the example 24 boards from work order number 13 are held in the incoming queue.

the flow of assemblies through the operations over time. [7] A sample of a "Spencer Diagram" is shown in Figure 7. The various operations are arranged along the vertical axis and data collection times are shown across the horizontal axis. Numbers inside the diagram show how many boards were found in each operation. Lines are drawn between pairs of numbers to show the flow of the boards during the previous four hours. The slopes of the lines indicate the relative speed of the flow; steeply sloped lines indicate fast movement through the process. In Figure 7 quantities of boards in queue are noted in triangles; quantities of boards being worked on appear in circles.

Analysis of the data showed that most of the time boards were in queue waiting to be worked on; in fact, in the initial analysis, no boards were ever observed in several of the short-duration operations. The data showed that the entire lot of boards often remained in the initial queue for several days before the first operation was started. It was found that when parts were on back order, no work was performed until the kits were complete, except in cases where the work was late. It was also noted that in one case boards were moved from one queue to another with no operation in between! It could also be seen that although most lots consisted of from 30 to 70 boards, the production workers broke each lot into small quantities of from 4 to 12 boards; these small sub-lots tended to move through the process at varying rates. In several cases sub-lots moved backwards indicating some abnormal condition in the process.

Next, actions were started to make improvements in the process. First, an analysis made of the causes of back orders

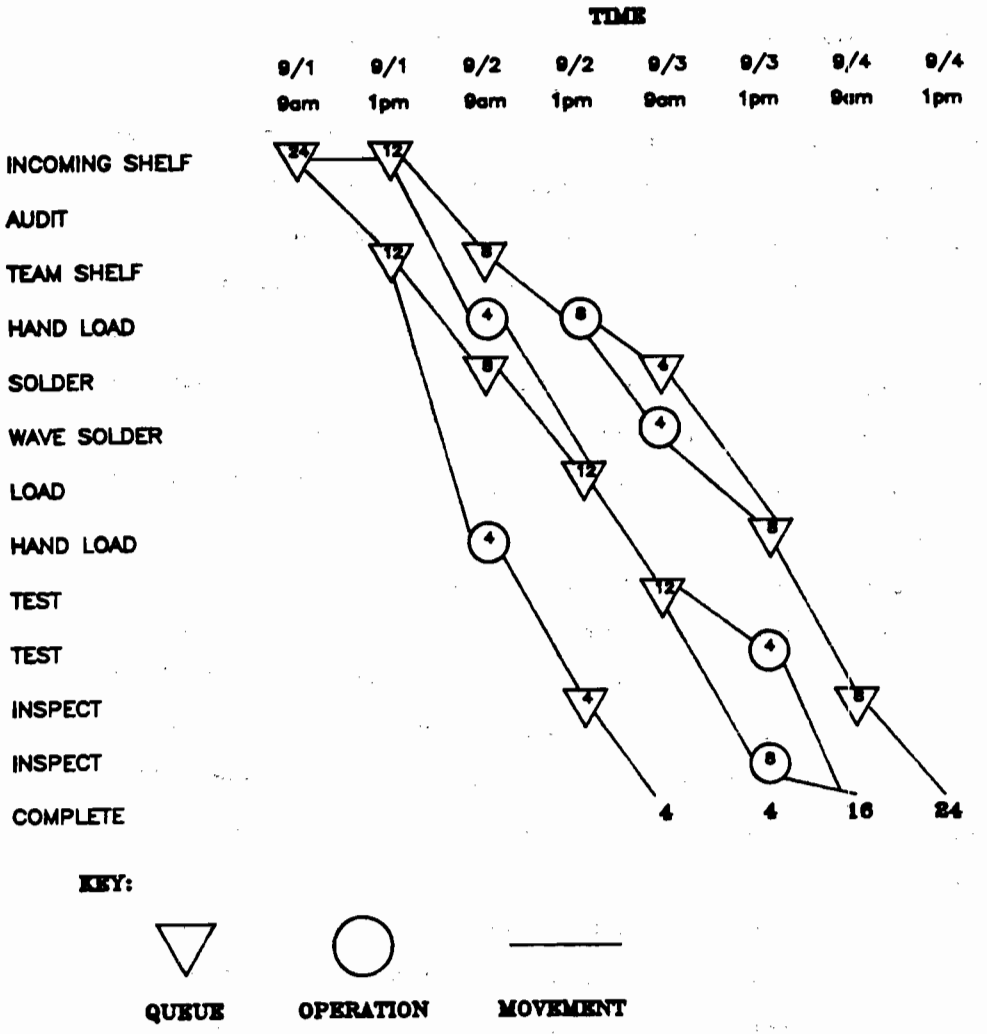


Figure 7. "Spencer Diagram". In this example the boards from this work order were first observed in the incoming queue at 9 a.m. on 9/1. By 1 p.m. on 9/4 all boards had been completed. The total boards observed at each time are equal to 24. Data collected in this manner over many work orders can be summarized to yield an accurate picture of how material flows through the process.

showed that a problem existed in the procedures used for incoming test of random access memory (RAM) IC's. It was found that schedules for this area were not linked directly to the MRP scheduling system. In many cases it was not known by the incoming test department exactly when RAMS were needed by the assembly area so deliveries were often later than expected. It was also found that the IC buyers were not aware of the special procedures which meant that RAMS were often received from vendors too late in the cycle. New scheduling procedures were created and explained to the buyers and the problem of late RAM's was eliminated.

To prevent shortages of other components, a look-ahead procedure was implemented by purchasing so that potential back orders were made visable and expediting could be started earlier. With these new procedures in effect, the average cycle time was reduced to 5 days; the MRP cycle time was then reduced and a significant decrease in work-in-process (WIP) inventory was achieved.

After the reduction in cycle time was achieved, collection of operation data was discontinued; it was then observed that cycle times for the assembly increased by several days, but remained at a level still significantly lower than before. Investigation revealed that while data was being collected the production workers were expediting boards through the process; once data collection was stopped, the workers stopped expediting. The workers were instructed, that when further data was to be collected, no expediting was to be done and boards should be allowed to flow normally through the process.

The results of the initial actions taken to reduce cycle times were summarized and shown to all supervisors and process engineers in the assembly department. Encouragement was given to conduct further studies of the assembly process and to try for further improvements. The group was challenged to work toward reducing average cycle time to less than 24 hours. This aggressive objective was deemed feasible since newly installed wave solder and automatic insertion equipment had increased capacity sufficiently to allow queue reduction in these critical areas.

The new objective for cycle time reduction spurred process improvement activity in all areas of the assembly area. One group suggested that if only complete kits were allowed into the process, significant time could be saved by the production workers. It was pointed out that many lots of boards were partially assembled and then placed on shelves awaiting the arrival of missing parts; in some cases the assemblies were worked on and shelved several times as missing components were gradually received. Another compelling reason to adopt this suggestion was provided by the wave solder team leader who reported that the majority of solder defects for the prior month were found on a lot of boards which had been shelved for several weeks awaiting parts. Possible explanations of increased solder defects were that oxidation was occurring on exposed component leads or that moisture absorbed from the air was interfering with the soldering process.

With all in agreement, Stores was instructed to change their

procedures such that only complete kits were delivered to the assembly area. The effects on the assembly process were both dramatic and immediate. A large stack of incomplete kits was formed near the stores area and a tour of this area was arranged for the buyers so they could see the full impact of late materials; they were asked to cooperate with the assembly area to help solve this now visible problem. As incomplete kits were gradually flushed from the assembly area numerous shelves became empty and were dismantled and removed from the area. Several hundred square feet of free space appeared in the assembly area. An immediate improvement in morale and productivity was observed as significant complexity was removed from the assembly process.

The addition of appreciable space allowed several process changes to be tried. A number of operations were combined and workstations were relocated so that less material movement was required. Assembly teams were reorganized so that lots of boards could continue flowing through the process on both day and swing shifts; previously each team had been assigned certain assemblies which meant each type of board could only be worked on during certain hours of the day.

As work-in-process queues were eliminated a problem surfaced: fluctuating amounts of work delivered to the beginning of the process caused frantic activity during some periods and complete inactivity during other periods. [8] The problem was studied by Production Control and the decision was made to reduce lot sizes significantly so that lots of each board type would be delivered several times each week or even daily in the case of high volume assemblies. It also became apparent that the new

assembly process could not tolerate significant down time of critical equipment such as the automatic insertion and automatic test equipment. A study was begun to work on ways to reduce the average repair time of each piece of equipment.

The data now showed that a significant reduction in cycle time had been achieved. With the reduction in lot sizes not yet in effect the average cycle time appeared to be in control with a mean of 5.5 days, down from 16.35. Sorting the data by lot size showed that further improvements would be achieved as smaller lots reached the assembly area.

Results and Conclusions.

At the beginning of the project the primary objective of reducing cycle times was to reduce WIP. Although a reduction in WIP was achieved, the unanticipated positive effects should be emphasized. In addition to improved productivity, morale, use of space, and quality, other small gains were noted. The requirements for cardboard tote boxes, mylar protective bags, transfer carts and other supplies related to the amount of WIP were all reduced significantly. The goal of 24-hour cycle time now seems within reach and should be achieved with 6 to 9 months of continued effort.

DISCUSSION OF THE KEY FACTORS NECESSARY FOR SUCCESSFUL IMPLEMENTATION OF STATISTICAL QUALITY CONTROL

Introduction.

The experience of implementing Statistical Quality Control at CSY has made it clear that there are a number of conditions which are favorable to accelerating the implementation process; the absence of one or more of these conditions may hinder the program or halt it completely causing much discouragemnt for the work force. As each point is discussed, an attempt will be made to outline specific actions that we have used in our organization to provide a more fertile climate for growth of an SQC program.

Top Management Must Clearly Understand the Need for Improvement.

Our experience supports the claims of W.E. Deming that if quality is to be improved, top management must feel the need to improve quality and must communicate this feeling to all levels of the organization in order to establish a proper climate for SQC implementation. Gaining an understanding of the rapid progress being made by the Japanese in improving quality should help management internalize this need and feel a sense of urgency.

A prime example of how to get this point across was presented recently at a lecture given by several managers from the HP division based in Japan. The subject of the lecture was a discussion of the program of Total Quality Control (TQC) in use there which had helped them win the coveted annual Deming Prize for quality improvement. The managers were explaining the use of statistics in improving the order closing ratio of their salesmen

and used the term "Attack Targets" several times in the presentation. It soon became clear that this term referred to potential customers with whom the salesmen were working! Japanese companies are waging an economic war to expand their market share and are surely studying new markets to penetrate in the United States.

Today, many competent lecturers are available for hire who can present the facts to management; Deming, Perry Gluckman, Philip Crosby, Juran, and William Conway are several who can help establish an urgent need for action in the ranks of top management. Urgency for action must also be communicated to every employee in the organization; it should be stressed that top management will be prepared to provide the proper tools and other incentives necessary to implement a quality improvement program. A detailed implementation program is presented in Crosby's "Quality is Free" and other examples are available.

Management Must Take an Active Role.

Two kinds of support are required from management to implement a successful SQC program. The first, which can be labeled "passive support", can be described as the support necessary to allow the program to proceed. Management must make available competent statistical help who has a thorough understanding of the concepts of SQC and how they can be applied for process improvement. Also, a sufficient budget to cover consulting services, training supplies and materials, meeting rooms, and other needs must be provided. Management must "bless" the program and let the work force know that they support the

program and that rewards will be provided to those who are successful in achieving improvements. Passive support is necessary, but not sufficient, to implement an SQC program; active support must also be given generously.

To provide active support a manager should be as thoroughly trained in the principles of SQC as any of his staff and should actively participate in projects as a member of a team. The manager should take an active role in leading training sessions. The manager must show a high level of enthusiasm and ensure his subordinates demonstrate this also. The manager should encourage everyone to circulate copies of control charts, Pareto diagrams, minutes of meetings, and other documents; he should personally make positive comments to the originator of such materials every time they come to his attention.

The manager should make sure that control charts and other quality measures are prominently displayed at every point in the process where data is being collected and projects are being worked on. A good procedure is to walk through the plant several times each week, look at the data that is posted, and talk informally with the work force so an understanding of the problems being faced can be gained.

The manager must be prepared to take immediate action to remedy causes of problems when appropriate data is presented to him. The manager must provide an atmosphere that encourages the workers to identify and report problems without being criticized. The manager must make sure the workers know that he understands that most problems are part of the production process itself and are not the fault of the workers.

One major reason for the success of the program at this division is the heavy involvement managers have in the program. After one year in the process a survey was conducted to measure the number of SQC tools in use and the results are shown in Figure 8. Most managers and supervisors had used two or more SQC tools and nearly three-quarters of them had employees who had also used tools.

Figure 9 shows in more detail the strong motivation of the employees to adopt the use of SQC tools if the supervisor was also using them. Few employees use tools which are not used by their supervisors; most supervisors who have used particular tools have employees who have also used those tools. It is suggested that each manager undertake some project of data collection and analysis to become familiar with the tools available in order to set a good example for his people.

Training Must Be Ongoing and Thorough.

Learning to be proficient at the business of SQC is much like learning to become expert at the game of golf. Just as it is inappropriate to expect a novice who hits long accurate shots off the practice tee with a competent professional at his side to achieve par under course conditions, is it unwise to expect to see immediate and major improvements in quality after giving a two-day course to the work force on how to construct control charts. A long-term program of training must be developed and the concepts be practiced diligently.

The first step in the training should be a short course in basic SQC concepts presented to managers down to the supervisory

Number of SQC Tools Used	Number of Managers
0	1
1	1
2	5
3	3
4	6
5	3
6	4
Total	23

Figure 8. Number of SQC Tools Used by Managers. Of 23 managers surveyed, 22 had used at least one tool and 4 managers had used all six tools covered in the survey.

	Managers Who Have Not Used This Tool		Managers Who Have Used This Tool	
SQC Tool	Whose Employees Have Not Used This Tool	With Employees Who Have Used This Tool	Whose Employees Have Not Used This Tool	With Employees Who Have Used This Tool
Process Flow	7	-	6	10
Cause-Effect	10	-	3	10
Pareto	6	-	4	13
Histogram	12	-	3	8
Control Chart	3	2	6	12
Scatter Plot	13	2	4	4

Figure 9. Relationship Between The Use Of SQC Tools By Managers And Their Employees. The survey data shows that few employees use tools not used by their manager. The data also shows that a majority of managers who used a particular tool had at least one employee who had also used that tool.

level; other professionals including process engineers should also attend the introductory training. After training the students should be able to construct simple types of control charts and should be able to do some interpretation of the data. The students should also gain a basic understanding of probability theory; the Brown Bead Company Exercise (See Appendix B) has been proven to be an effective method of conveying simple probability concepts and demonstrating the difference between process problems (that can only be removed by management) and problems that the workers have some control over.

Next, participants should be encouraged to pick some condition that they would like to see changed and begin collecting data so a process can be studied; the first attempt should involve a simple process in the working environment such as measuring machine output or counting defects.[9] Examples from the students' personal lives can also be used effectively such as measuring daily body weight, daily commute time or automobile gas mileage. Extensive help may be necessary at this early stage of training to ensure participants are successful.

As the implementation process proceeds, a long-term program of training should be begun so that the initial high level of motivation of the workforce can be maintained; without continuous training, any member of the staff who experiences difficulty with a project may become frustrated and decide to abandon using SQC. An approach to this type of training which has been used successfully at CSY will now be described.

After several months of the implementation process a number of successful projects had been completed as described earlier.

The statistical consultant continued to come to the plant one day each week and anyone who wanted help could arrange for a consultation by reserving time on the consultant's desk calendar. However, it was becoming apparent that the number of active projects was not increasing and was evidenced by the small number of people signing up to talk with the consultant. In order to stimulate further projects the department manager announced that an SQC study group was to be formed and that all supervisors, section managers and process engineers were invited to attend.

It was decided that the study group would read SQC source material together, discuss the concepts presented, and attempt to see if ideas and techniques might be relevant to the processes in the department. The first material selected was QUALITY, PRODUCTIVITY, AND COMPETITIVE POSITION by W. Edwards Deming. This text had been published in late 1982 and was designed to be used with an extensive series of video tapes of Deming's lectures. The group met for one hour each week to discuss what they had learned in the assigned reading. A list of discussion questions had been given the group in advance to stimulate discussion. The group was led by the department manager; the consultant and a resident statistician attended to help with the explanation of the more difficult topics.

As the group progressed in the reading, other materials were added to broaden the content. Video tapes on quality control, guest speakers and original material prepared by the statisticians were all used effectively. Also, each week one member of the staff who was collecting data was invited to show

the data and lead a discussion focusing on a description of the process being monitored, whether or not the process was in control, and what further activities were planned.

The weekly sessions often generated lively discussions as the staff discovered that many procedures then in use in the manufacturing area were challenged by statements in the readings. After some six months of classes the level of understanding of the SQC process had increased significantly as could be seen by the fact that some 30 processes were in some stage of analysis. However, even this extensive training had failed to provide nearly half the staff with sufficient skills or motivation to work independantly on projects. It was evident that some members, especially those with little college level training were having difficulty grasping some of the concepts.

It was decided to continue the weekly sessions for at least six more months in order to bring all members up to a common level of understanding. The next material selected for analysis was GUIDE TO QUALITY CONTROL by Kaoru Ishikawa. This is an excellent introduction to a number or simple statistical tools with good discussion on how to use them and numerous exercises which can be worked and discussed. It was also planned to introduce some material on Japanese manufacturing methods as continued process analysis would undoubtably lead to the use of some of these techniques.

In addition to the weekly classes another method of sharing information and increasing motivation was tried. All supervisors, managers, and engineers were invited to a meeting to listen to a discussion of what SQC projects were in progress.

Anyone collecting data was invited to share their experiences in front of this group. Two two-hour sessions were held with good attendance and were considered valuable in promoting SQC concepts to departments which were not currently working on projects. A further benefit was a chance to show management which of the processes needed to be changed and what problems were being encountered.

Positive Rewards for Success Must Exist.

In order to ensure successful implementation of SQC it is vital that positive reinforcement be provided to all participants at each step in the process. It is even more important that participants not be penalized for their activities. First we will undertake a discussion of positive rewards and get to the penalties later.

As is true in stimulating any behavior, positive reinforcement must be given by those in charge of the organization. It has already been pointed out that merely by observing actions taken by the workforce and making positive comments, workers will gain recognition and feel more positive about the process. Managers should also recognize improvements by giving public recognition to teams which have solved problems and by writing articles for company publications which give visibility to participants. Company-sponsored lunches or dinners with managers in attendance will provide long-lasting positive good feelings among the workers. Although workers will be justifiably proud of improvements they have made, there is no substitute for a personal and sincere thank-you from the boss.

Several characteristics of the SQC process can lead to the giving of severe penalties for those who participate in making improvements. In some cases management may be unaware of the negative rewards inferred by the workers for the actions they take. Obviously, workers are not likely to tell management how to do more with less if they or their friends will have to make a painful job transition as a result. People in incoming inspection may be reluctant to collect good data on the quality of parts if they see some of their co-workers forced to change jobs because past data they collected resulted in improvements in quality of incoming parts which reduced the need for inspection. Middle managers should not be rewarded by demotions if their improvements result in a reduction of the size of their departments.

Another problem is the potential loss of visability which can be suffered by a department which has made process improvements. In many organizations managers who are excellent at solving crisis situations receive the majority of top management attention while those who have smooth running processes receive much less attention. A quote from an article written by Charles Quackenbush illustrates this point. He says, "If you spill milk and then try to save it, you are an American results-oriented manager. If you change the process so there is less spilled milk, you are a Japanese process-oriented manager."[10] It is clear that the new way of managing will concentrate on preventing problems before a crisis happens.

A final potential problem is related to the characteristic of SQC analysis to expose previously concealed problems and to search out the causes; if this process causes a problem to be discovered in one organizational area, and the cause happens to be located in another, ill feelings may result toward the group which has exposed the problem. Forming a team to study the data with members of all involved groups before the cause is discovered can sometimes eliminate this problem and lead to better teamwork within the organization; if it is perceived that one department is "pointing fingers" at the other, the effectiveness of an SQC program will be severely reduced. Forming a team with representatives from a supplier who is thought to be supplying poor quality materials or making late deliveries can lead to substantial gains; cooperative study of the data may lead to the discovery that the specifications are in error or the customer has been ordering within lead times.

CONCLUSION

Implementation of SQC at the Computer Systems Division has led to significant improvements in both quality and productivity. We have discussed several case histories of successful projects and tried to explain some of the reasons for the successes. We have also discussed some of the key management factors that need to be present in an organization to stimulate growth of the concepts of SQC.

Proposal for Future Quality Improvements.

We will now propose a three-step program, based on our experiences, which should lead to further gains in quality and productivity.

1) Ensure a Participative Work Environment. A positive environment must be provided for the work force which encourages creativity in analyzing and improving processes.[11]

2) Institute A Program Of Company-wide Quality Control With SQC As A Driving Force.

3) Adopt A "Zero Inventory" Philosophy In The Manufacturing Process.

We feel that a participative work environment is necessary to realize large gains in quality and productivity. Employees must not be afraid to report problems and collect data and must be assured that management will take actions to improve the work. Once that climate is established, SQC and Zero Inventory programs can be used together to produce good results.

From studying the results of these programs at HP and other companies it appears that Zero Inventory programs can achieve good results but must be integrated with a company-wide quality program so that full benefits can be realized.

NOTES

1. For convenience it was decided to measure the number of defects solder joints per million solder joints produced. In this example, and in later ones to follow, defect rates will be referred to as "Parts Per Million" (PPM). This notation was found to be easier to work with than very small percentages. For example, 6 PPM is easier to work with than 0.000006.

2. This is one of the many examples we found of the ineffectiveness of inspection in eliminating errors and defects from production. Deming describes many problems with inspection. He advocates inspection for process surveillance and control, but notes that inspecting to screen out defects is a relatively ineffective and expensive way to obtain quality.

3. In attacking inter-departmental problems the importance of agreeing on a common performance measure must be emphasized. Also, it is preferable the the "customer" or receiver of materials or assemblies have the major say in determining the proper measure. In the material discrepancy case mentioned here, the Stores department measured their performance in terms of the percentage of pulls in error and since their error rate was less than 1%, they considered their performance to be very good. However, since each workorder kit had many part types, nearly two-thirds of all kits received by the assembly area contained at least one discrepancy; clearly, then, this was a situation that needed improvement.

4. For a thorough treatment of cause-and-effect diagrams see Ishikawa, Chapter 3.

5. Note the difference in the use of the term "cycle time" from its meaning in the Toyota Production System and in much of the current literature describing just-in-time production or zero inventory systems. In these cases cycle time refers to the length of time between completion of similar items in the production process; it may be expressed as "25 bearings per minute" or "a car every 72 seconds".

6. In order to construct accurate process diagrams it is important to carefully interview line supervisors and production workers in order to include the informal queues and operations which have been developed to handle production problems encountered in the process. Changing the process without considering the informal operations may not lead to the desired results.

7. I am indebted to Spencer Graves of the Santa Clara Division of the Hewlett Packard Company for constructing the first of these diagrams I have seen which has made the analysis of material flow through complex processes quite simple.

8. As process changes are made which reduce intermediate WIP queues, problems of uneven material flow will be exposed causing production areas to be lightly loaded for varying periods of time. During these periods management should reduce the emphasis on measuring labor variances and machine loading so that attention can be focused on leveling the work flow. In many cases reducing queues will reduce process complexity to improve productivity more than enough to offset idle labor and machine time.

9. See Appendix A for a complete description of the process improvement process.

10. See the Quackenbush article.

11. See Ouchi for a good discussion of the factors necessary to provide this environment.

12. For a thorough discussion of "Zero Inventory" concepts see Zero Inventory Seminar Proceedings.

BIBLIOGRAPHY

ANSI. Guide for Quality Control and Control Chart Methods of Analyzing Data. ANSI Publications Z1.1 and Z1.2, 1958.

Control Chart Method of Controlling Quality During Production. ANSI Publication Z1.3, 1959.

APICS. Zero Inventory Seminar Proceedings. APICS, September, 1983.

Crosby, Philip B. Quality is Free. 1980.

Deming, W. Edwards. Quality, Productivity, and Competitive Position. MIT Center for Advanced Engineering Study, 1982

Gluckman, Dr. Perry. Introduction to Statistical Quality Control. PC Fab, March, 1983.

Managing Vendor Relations. PC Fab, April, 1983

Using Statistical Quality Control to Improve Performance. PC Fab, May, 1983.

Adjusting Data to Use Control Charts. PC Fab June, 1983.

How to Detect, Isolate and Solve Problems Using SQC. PC Fab, July, 1983.

SQC Vs. The Same Old Way. PC Fab, September, 1983.

Grant, Eugene L., and Leavenworth, Richard S. Statistical Quality Control. McGraw-Hill, 1980

Ishikawa, Karou. Guide to Quality Control. Asian Productivity Organization, 1976.

Juran, J. M., and Gryna, Frank M. Jr. Quality Planning and Analysis. McGraw-Hill, 1980.

Monden, Yasuhiro. Toyota Production System. Institute of Industrial Engineers, 1983.

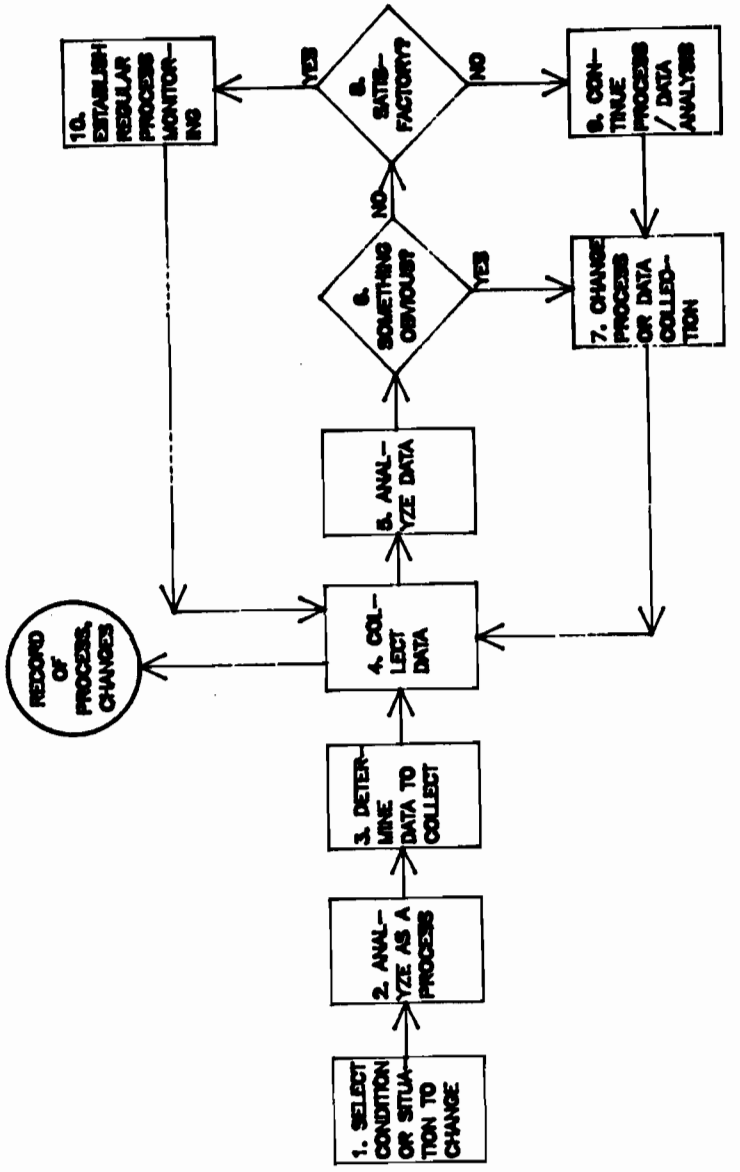
Ouchi, William. Theory Z. Addison Wesley, 1981.

Quackenbush, Charles. A Change in Management Techniques Could Keep Jobs Here. San Jose Mercury, March 6, 1983.

Schonberger, Richard J. Japanese Manufacturing Techniques. The Free Press, 1982.

APPENDIX A

PROCESS FLOW DIAGRAM FOR AN SQC PROJECT



Process - Flow

for an

SQC Project

1. Select a situation or condition you want to change.
2. Analyze the operation as a process,
   a. Informally.
   b. Make a process-flow diagram.
   c. Make a cause-effect diagram.
   d. Hypothesize some other kind of model (e.g., relational diagram, algebraic model).
3. Determine what data to collect.
   a. What is unknown that might to useful to know?

	Quality	Quantity	Timing	
Inputs				
Intermediate				
Outputs				

   b. What variables could be measured? What qualities could be quantified or counted?
   c. What could we do with such data if we had it?
      (i) Pareto Diagram
      (ii) Histogram
      (iii) Control Chart
      (iv) other

d. How could we reliably collect such data to get the information we want with minimum work?

(i) Where in the process?

(ii) Using what test or measurement equipment or what kind of qualitative standard? (For example, what constitutes a defect? And how could calibration of test equipment or qualitative standards be maintained?)

(iii) In what format or using what kind of data collection form?

(iv) Who would collect it?

e. Is 100% data collection feasible? If we sample, how do we select the (random) sample?

f. Can we do something simple and sensible now, and change it tomorrow or next week depending on what we learn? (If yes, do it. If no, we must be more careful in thinking about what we want to know and how the data we collect might help us.)

g. Is such data collection feasible?

(i) If no, go back to substep 3.a.

(ii) If yes, proceed to step 4.

4. Collect data to monitor the process over time. Maintain a log of all changes made to the process and the impact these changes had on the data.

5. Analyze data: Make a Pareto diagram, a histogram, a time series plot, a control chart, or ... .

6. If some obvious and easily corrected problem is apparent (e.g., a Pareto analysis or a histogram reveals something obvious, or a control chart has a point or points "out of control," and the reason is obvious), go to step 7. Otherwise, go to step 8.

7. Change the process or data collection. If the indicated change requries the approval or cooperation of others (supervisors, managers, or coworkers), present the change and the data and analysis for their review. Continue with step 4 to see if this change has any impact.

8. Is the Process satisfactory?

a. Is the defect or rework rate too high (relative to the other difficulties we face?)

b. Does a high percentage of the product meet specs?

c. What is the "actual manufacturing cycle time" relative the the scheduled cycle time?

d. What percent of the actual manufacturing cycle time is queue time?

e. How much idle inventory is being maintained?

f. What percent of the time is equipment down for unscheduled maintenance?

If the process is satisfactory, make sure appropriate monitoring is maintained to keep it that way, as described in step 10. Otherwise, continue with step 8.

9. The process is not performing satisfactorily:

a. Continue the process analysis begun in step 2.

(i) Do the data help us isolate a problem? Do they suggest a potentially benificial change in the process? Can that change be implemented at least on an experimental or temporary basis to see if it actually has the desired impact on the process and the data we collect?

(ii) Would a change in the data collection procedures provide better information to help isolate a source of difficulty? Or could the data collection procedures be changed to provide the needed information with less work?

b. Set upon some change either to the process or to the data collection methodology, and continue as in step 7.

10. The process was found to be satisfactory at the moment. To make sure it stays that way, proceed as follows:

a. Review alternative sources for data that would provide reasonable information on the status of the process and would still be easy to collect.

b. Establish (or continue) procedures for regular process monitoring using a control chart based on these data.

c. Continue with step 4.

APPENDIX B

The Brown Bead Manufacturing Company *

Materials: A box containing several thousand small, light-colored beads and a few hundred red beads of the same size. A paddle (See Figure 1) which can be used to draw samples of 50 beads at one time.

Start the exercise by announcing that all participants have been hired to work for the Brown Bead Manufacturing Company, whose primary organizational objective is the manufacture of small brown beads. Show a sample of the company's product line and point out its features to the participants.

Pass out one blank control chart to each participant. You should also have had an overhead transparency made of a blank control chart for your use during the exercise.

Have each participant draw five samples of fifty beads. As each draw is made, have all participants record the number of red beads drawn in the appropriate spot on their control charts. For the purpose of this exercise, approximately 20 sets of data points (100 draws) should be drawn and recorded. As participants record the data points on their control charts, you should also record the data points on the overhead transparency. Interrupt the drawing procedure several times as follows:

1. After the very first draw is made, stop the participant and say something like, "I notice that you have drawn several red beads. How do you feel about that?... Did you know that the objective of this company is to manufacture brown beads, not red beads?... Well, listen. I think that this is probably my fault. I didn't make sure before we started that the objective was really clear to you. But now that we've clarified and reached agreement on the objective, I would look for an improvement in your performance. Do you think you can achieve that?"

Learning Point: Managers can't assume that simple because they have clarified objectives for their employees that they will be more motivated or more able to accomplish a task.

2. After several participants have drawn beads, stop after someone has had a high draw of red beads. Indicate that this is unacceptable and ask a participant who has already drawn and who had a low average of red beads to provide training. Ask the

* This exercise was developed from work done by W. E. Deming, Bill Boller of the Hewlett Packard Company, and Dr. Perry Gluckman.

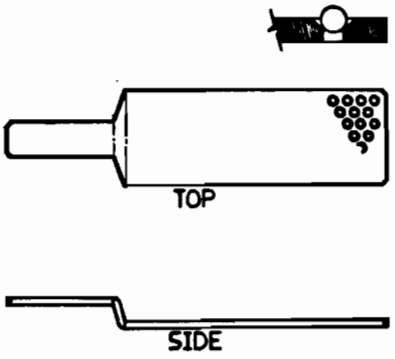


Figure 1. Bead Paddle. The illustration shows a sketch of how a paddle could be constructed. Drill holes slightly smaller than the size of the beads.

participant to show how he or she holds the paddle, how deeply to draw, what motion to use, etc. After training, commit the participant who had drawn a high red bead count to improve.

Learning Point: Training may be powerless to resolve problems which reside in the system.

3. Next, indicate to another participant before he or she begins to draw that you are going to make his or her pay contingent on performance. Offer the person $5 if he or she can achieve an average of "n" (n should be selected such that its probability is .01 or less.) red beads or less for the next five draws.

Learning Point: Pay can also be powerless as a motivator when systems problems exist.

4. After another participant has made two or three draws, threaten to fire him or her if the person draws over three (or some number) red beads in any of his or her remaining draws. Indicate that you're simply going to have to make an example of him or her for producing too many defectives. If more than three red beads are drawn, fire the participant.

Learning Point: When there are problems in the system beyond the control of the employee, management pressure can only be counter productive. Emphasize that 85% of the production process problems that surface are attributable to problems in the system. (Common causes) Only the remaining 15% are caused by employee error. (Special causes) This is Deming's rule of thumb

and as he points out, the burden for removing the systemic problems is clearly on the shoulders of management, not the employees.

5. Next, after most of the draws have been made, ask a participant to predict how many red beads will be drawn on each of the next two draws.

Learning Point: No matter how much data is collected, the exact number of red beads cannot be predicted with certainty.

6. Before the next participant begins drawing, ask someone to predict the average number of red beads for the next five draws.

Learning Point: Since each hole has an equal chance of drawing a red bead, the average number of red beads in a draw tends to cluster around a certain number. (This is the central limit theorem).

7. Continue drawing beads and recording the data until approximately 20 sets of data points have been drawn and recorded on the overhead and on participants' control charts. Next, talk about control limits. Ask the group to pick a range within which they expect the average of the next five draws to fall. Try to get the narrowest estimate of a range in which they would feel confident that the average of the next five draws would fall. Then tell them that control limits can be established showing a range within which 99.7% of the averages will fall when the process is in control. The control limits are derived by establishing a center line which is the average of all the averages plotted and then by moving three standard deviations in both directions away from the center line.

Learning Point: Setting control limits can define the capability of the process and serve as an indicator of when the process is out of control and the cause investigated.

At this point have the group plot each of the data points on their control charts, plotting both the averages and the range. Then as a group, calculate the upper and lower control limits for both the X-bar and R parts of the chart. Then review the main points made during the exercise.

Now shift the discussion to the participants' on-the-job situation. Ask participants what they consider to be the red beads in their operations. These might include poor parts from a vendor, late deliveries, unclear standards, machine breakdowns, etc. Record the group's ideas on a flipchart and facilitate discussion until all ideas have been shared.

LOCAL AREA NETWORKING: ISSUES AND ANSWERS

JIM GEERS
SYSTEMS NETWORKING SPECIALIST
HEWLETT-PACKARD COMPANY

Introduction

Probably everyone in the computer industry agrees that by the end of the 1980's, there will be a powerful computer-based workstation on your's and everone else's desk. To illustrate my point, think back 10 years ago when the pocket calculator was just becoming popular. Because of its costs though, the pocket calculator could only be cost-justified by certain specialists such as Accountants and Mathematicians. As we know, advances in technology have allowed us to put this capability into everyones' hands today. Computer-based workstations are at the same stage as pocket calculators were 10 years ago and so it is only a matter of time until technology makes this capability available to everyone.

Technology, however, isn't the only factor driving this trend in computing. Today we're seeing an explosion in computer-based services such as data base libraries, office systems which include word processing, electronic mail and graphics, and computer-based applications which assist the specific profession or job at hand. Therefore, the other key factor is the need to increase the productivity of each individual within your organization by providing access to these information services and resources. This will make the computer-based workstation a tool as necessary as our telephone is today. As Joel Birnbaum, Director of HP's Computer Research Labratories, states, "The future belongs to machines that attach to your telephone ... success will depend on communications links, not computer technology".

Recent studies have shown that up to 80 percent of all communi- cations traffic takes place within a building or local complex. And that, of course, underlines and emphasizes the requirement for local area networks.

Strictly speaking, the term "local area network" is a generic term which includes any interconnection device that is situated in a local environment and which connects together information processing equipment. A "local environment" may be a building, or a complex of buildings.

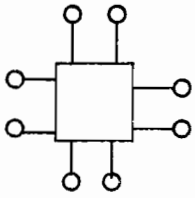
The other term in the phrase is "network". A network may be as simple as one system connected to many terminal and peripherals. However, the term "Local Area Network" (LAN) usually refers to more than two intelligent devices which are logically connected together and which can transmit information back and forth. The intelligent devices may be mainframes, minicomputers, personal computers, workstations (intelligent terminals) and other peripheral resources.

Other characteristics of LAN's are that they are generally owned and operated by a single organization, its distance usually does not exceed 6 kilometers, and it is tailored for maximum performance in a limited area. In other words, it trades long distance capability for high speed and transmission accuracy.

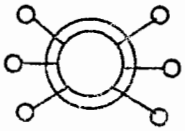
What is NOT a local area network? An Anaheim-to-New York connection is not local. There are methods to communicate remotely, such as Public Data Networks, leased lines and dial-up lines. A local network can be connected to a remote environment using these transmission methods.

When considering a local area network, what are the key differentiating factors? There are several factors involved such as access methods, capacity, reliability, security, etc.. The primary differentiating factor, however, in providing a network to improve your organization's access to information is topology. The three types of network topology are shown below:

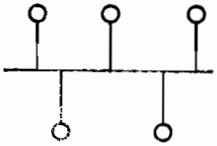
NETWORKING TOPOLOGIES



STAR



RING



BUS

The "star" network is primarily implemented today in the form of a Private Branch Exchange (PBX). This network is important because it exists in every building today in the form of twisted pair wiring attaching your telephones to a central switching device. PBX's are considered a type of local area network although the primary form of information transmitted today is voice. Recent developments in PBX technology now allow data to be transmitted as well.

The "bus" and "ring" networks are recommendations for what the industry has been calling "Local Area Networks". These "cable" based networks have taken a competitive position to the digital PBX as the central communications hub. One of the major advantages is that these networks represent the beginning stages of universal connectivity for all information processing equipment. LAN's were primarily developed for data transmission however new developments make it now possible to transmit other forms of information such as voice and video signals.

Local Area Networks Vs.
Private Branch Exchanges

Because all of these networking concerns are evolving to common capabilities of handling all forms of information, which one is best suited for your organization's needs? here are no hard and fast rules today to determine whether an LAN or PBX is right for your organization, but here are a some points to consider:

- o) If your organization's PBX equipment is old and you see significant addition voice requirements in the years to come, a new digital PBX with data commuications capability may be less expensive than purchasing a new voice system and adding a LAN at the same time.

- o) If the number of systems to be connected are small and there is no immediate need to replace existing voice systems, a new digital PBX system will probably not be economical. New PBX systems must still be cost-justified based on its voice capabilities today. Costs are still relatively high to connect data processing equipment.

- o) Where very high data speeds are involved (over 56 Kbps), will be difficult to find a PBX today supporting higher speeds. An LAN would be the best solution.

- o) If you have many terminals to connect to a system, or especially, many systems (i.e. office application) PBX offers

higher overall bandwidth, switching and ease of interconnecting (via existing telephone wiring).

o) LAN economy is most obvious when the users are clustered in computer or terminal rooms throughout a facility. Office automation users are not necessarily placed so conveniently.

o) It is easier to supply centralized services such a gateway access, security, etc. through a centralized system such as a PBX, rather than a more distributed LAN. Of course, by relying on a single system, your risk of downtime is much greater. Therefore, the choice will depend greatly on your application.

Local Area Network Trends

As mentioned earlier, technology is evolving to make these choices simpler and more economical. There are several new developments which should be available in the near future:

1) Further integration of computer and PBX systems will make the PBX alternative much more cost-effective

2) New switching systems will front-end the older archictectur PBX systems offering a virtual voice/data network. These new switching systems will allow you to front-end your PBX, splitting voice and data signals to the respective PBX and computer processing systems. This will give you all of the major advantages of a digital PBX system without the huge intial investment since you will be leveraging off your existing voice network.

3) Vendors are developing interfaces which allow the PBX and LAN to link together. This hybrid approach will allow you to select the best networking topology for you application while still being able to integrate all of you information systems into a single network.

Conclusion

As we all know, it is a very competitive environment today and it will become even more so in the future. You know that the computer industry has and will continue to develop the tools to help you manage all forms of information and make all areas of your business more productive. What you might not have considered is that within your industry, major competitors will be on the same level of computer automation in such key areas as accounting, manufacturing, computer aided engineering and office automation. Therefore, one of the key differentiating factors which will distinguish the leaders from the followers, is how effectively you exploit networking technology. Every effort today in planning to integrate all your information systems will put you into a better competitive position tomorrow.

James H. Geers Jr.

Jim is currently a Systems Networking Specialist for Hewlett-Packard's Business Development Group. His responsibilities include marketing of HP's networking products, articulating their networking strategy and collecting customer input for future product development.

Jim previously was a Regional Marketing Engineer, consulting with specific sales regions to assist them in achieving their sales objectives. Previous to HP, Jim worked as a Software Developer for Two Pi, Inc. of Santa Clara, a maker of IBM 370 plug-compatible systems.

Jim graduated from the University of California at Berkeley with a Computer Science degree.

CAP=PM; Privileged Mode De-Mystified

Jason M. Goertz
System Specialist
Hewlett-Packard

Introduction

Many years have passed since the first HP3000 rolled out the doors at Hewlett-Packard. In that time, an increasingly sophisticated user and software supplier base has emerged. Much of this increased knowledge and sophistication is because of better Computer Science education, as well as the fact that there are more and more people who have more and more years working with the HP3000. Along with this experience has come an increasing use of one feature of the HP3000, that being Privileged Mode.

In spite of this growing sophistication, there is still a large number of people who do not understand what Privileged Mode is. Even the in-house Data Processing departments writing PM code internally and software vendors who are supplying applications using PM (as Privileged Mode will be called from here on) do not always fully understand its consequences and dangers. It is the purpose of this paper to present a description of PM and its implications, and to provide a technical document that describes, in one place, all (or as many as could be found) of the various commands, compiler options, and intrinsics dealing with PM. It is a fundamental part of human nature to be fearful of the unknown. This paper is an attempt to make the largely unknown world of Privileged Mode known, and thus lessen the fear of this potentially powerful tool.

Before starting, some disclaimers are in order. This paper is by no means intended to encourage anyone to use Privileged Mode. It is the opinion and experience of the author, and does not constitute an official statement on the part of Hewlett-Packard. It is intended strictly to be informative so that the reader could form his/her own opinions.

The approach which will be taken in describing the details will be to look at PM from the inside out. That is, examine first what PM is at the hardware level, then look outward to the software which sets and checks for PM.

What is Privileged Mode

At the lowest level, PM is a state in which a process can run, either temporarily or for the duration of the process's execution. While in this mode, the ability to execute a certain set of privileged instructions is given, as well as allowing certain non-privileged instructions to operate in a different fashion than normal, user mode. In addition, procedures can be defined which can only be called when the calling code is running in PM. It is that simple. That is the total definition of PM. No magic, no wires or mirrors, no potions or incantations. However, there are many, many ramifications of this simple definition. Indeed, exactly what these instructions and capabilities are, and how PM can be entered and exited is the whole point of this paper.

The ultimate enforcement of which capabilities will be granted is done by the microcode (ie, hardware) of the machine. To understand how this is possible, we must first understand a little about the hardware architecture of the HP3000.

The CPU of any HP3000 has several hardware registers that are used for various functions. Different CPU types (Series

30,44,64, III) have different registers, but some registers are common to all types. One of these registers is called the Status register, and this contains information regarding the state of the hardware and microcode at a given instant in time. Such things as what code segment is being executed, whether an overflow has occured, whether carry has occured, and other types of information are kept here, and are available to both the hardware and software via this register. Bit zero (highest order bit) of the 16 bits is called the Mode bit, and this is where the fact that the machine is in PM or not is kept. If the bit is on (a one), then the machine is running in PM AT THAT INSTANT IN TIME. The microcode can check this bit at any time to determine whether certain operations are valid. The most common check that is made is when certain instructions are executed which are deemed "privileged". The list is too numerous to mention here, but can be found by consulting the Machine Instruction Set Reference Manual, PN 30000-90022. In the description for each instruction, the various checks that are made are listed. For example, the LOAD instruction lists STOV and BNDV checks. These are, respectively, the STack OVerflow and BouNDs Violation checks. One of the checks which can be made is the MODE check, which is whether or not the Status register Bit 0 is 1. In general, all IO instructions, and any instruction which can reference memory outside of the users stack or code segments have a MODE check on them. A few of these are MFDS (Move From Data Segment), MTDS (Move To Data Segment), LSEA (Load Single from an Extended Address), HALT (Halts the hardware), and SIOP (Start IO Program on HP-IB machines). Rather the list the instructions, it is better to consult this manual on a specific instruction and check if PM is required. When a violation is detected, a trap is executed which produces the message PROGRAM ERROR #6: PRIVILEGED INSTRUCTION.

The other check made by the microcode is done by the PCAL (Procedure CALl) instruction. It is possible to define a procedure with the OPTION UNCALLABLE keywords. When this is done, the PCAL instruction makes sure that the MODE bit is set when calling this procedure. If it is not, a trap is executed which produces the message PROGRAM ERROR #17: STT UNCALLABLE.

It is important to note at this point that it is not the instructions themselves that are particularly dangerous. It is the use of them that can cause problems. Even then, it is usually only two things that cause problems. One is when data is MODIFIED outside of the user's domain. Very rarely does just LOOKING at data outside the stack or code segment cause problems. The other and perhaps more difficult problem to ensure does not happen is providing incorrect information to the machine instructions or uncallable procedures. Something as simple as giving a data segment number of zero to the MFDS instruction will cause a System Failure 16. Or worse, a non-zero DST number which is not currently used. A timing problem in the creation and release of a data segment can be disasterous. It is almost entirely this area which has given PM code a bad name. Even MPE (which runs ENTIRELY in PM) cannot escape this problem.

It must be pointed out that there is a great deal of data which can only be obtained via PM that is perfectly safe to access. An example is data in the PCBX area of the stack. This is an area below the DL register which contains all kinds of file control blocks, extra data segment information, and other things. Since the stack is always there when a process is running, it is very safe to access this data. However, since it is below DL, privileged instructions must be used to access this area. One routine which does this is the JOBINFO procedure in the Contributed Library. This procedure access the PCBX to obtain the Job/Session number. A new intrinsic in MPE-V, also named JOBINFO, will replace this contributed procedure. It is impor- tant to note, however, that the new JOBINFO will do essentially the same work as the contributed version. Which brings up a good point: There is really two kinds of Privileged Mode. They are: 1) The kind HP supports and, 2) the kind HP doesn't support. All this really means is that HP writes MPE and utilities in Privileged Mode and supports it. When someone else writes the code, HP doesn't support it.

Setting the Mode Bit

We have seen that essentially what constitues PM is the MODE bit is set in the STATUS register. But how does this bit get set? Asking this simple question opens a veritable can of worms. We will try to make a systematic analysis of all the various options available to perform this simple bit-twiddle.

At the lowest level, the MODE bit is set by only one machine instruction, that being the PCAL instruction. The decision whether to set or not set the MODE bit on the PCAL by the following rules: 1). If the CST or CSTX entry reflects the fact that this segment is Privileged (Bit 1 of word 0 of the entry), then set the MODE bit on. 2). If the segment being branched to is nonprivileged, but the

MODE bit is currently set, then keep the MODE bit set for the execution of the new segment.

Note that 2) above implies that CODE WRITTEN TO RUN IN USER MODE WILL RUN IN PRIVILEGED MODE. Typically, this will not pose a problem. However, it is a little known fact that ALMOST ALL BOUNDS CHECKING IS TURNED OFF IN PRIVILEGED MODE. An example is the SCAN statement in SPL. If the termination criteria are not found, then normally a bounds violation will occur. In PM, however, the scan will continue beyond the stack. Obviously, this can have disasterous implications if the code has not been well debugged.

The reason for leaving the machine running in PM can be understood if we examine the logic for the EXIT instruction. This instruction can set the MODE bit to zero, but will not set it to one. The logic choices are: 1). If the currently running segment is in PM and the segment being exited to is in user mode, then the MODE bit will be cleared on the exit, returning the machine to a user mode state. 2). If the machine is in user mode at the time of the EXIT and the segment being exited to is privileged, (as indicated in the Status Register saved in the Stack Marker at Q-1), then Privileged Instruction violation will occur. This is done to prevent a possible breach of security. It would be a simple matter for a user without any special capabilities to write a small SPL program to call a procedure (a PCAL) which set bit zero of Q-1 to 1 (the MODE bit). Upon EXITing, the program would be running in PM, and could look at any part of the system, including the directory to get MANAGER.SYS password, or any other part of memory desired. Because the EXIT instruction follows rule 2, above, we see how this possible loophole is closed. We also see why the PCAL instruction follows rule 2 in the previous paragraph. If the procedure should, indeed, be privileged, this is the simplest way to insure that the MODE bit is set correctly.

Continuing on our journey outward, it was mentioned that the CST or CSTX entry (logically the same) was checked for a PM bit. This bit gets set when the CST or CSTX entry is created. It is best to take each case, CST and CSTX, separately.

The CST table is a fixed length table (192 entries) which contains information regarding segments found in SL files only. MPE-V will change this around in detail, but for the purposes of this discussion, this description will suffice. Each entry is built by one of two software entities. On startup, INITIAL reads the system SL (SL.PUB.SYS) and creates a CST entry for every segment which is a SYSTEM, RESIDENT, or PRIVILEGED segment. Typically, MPE segments will always be SYSTEM and PRIVILEGED. Thus, each segment marked PRIVILEGED will have the PM bit set in the CST. After the system is up and running, and all MPE segments are loaded and CST entries created, the LOADER is the entity that adds any other SL segments to the CST. This is done when a program is loaded which references procedures in a segment in a group, public or the system SL. If that segment is also marked as being PRIVILEGED, then the PM bit is set for that segment also.

For segments that are resident in program files, the rules are the same as for user SL segments, above, except that table that gets built is the Code Segment Table Extension (CSTX) instead of the CST. Also, the information as to whether the segment is privileged or not is kept in record 0 of the program file in a table called the CST re-mapping array. Thus, when a :RUN command is entered by a user, the LOADER reads this table and knows whether or not to set the PM bit in the CSTX. It is important to note that the smallest entity which can be called "privileged" is a segment. This has some important ramifications. If one procedure in a segment is defined as privileged, then ALL PROCEDURES IN THE SEGMENT WILL BE PRIVILEGED. It is therefore a good idea to keep all privileged code together, and not mix the privileged procedures with ones that run strictly in user mode.

Continuing outward, we must ask how the program file or SL file is built. In each case, the answer is the same: the SEGMENTER. Whether used directly via the :SEGMENTER command, or indirectly via a :PREP command, the SEGMENTER is used to transform a USL file segment or segments into a finished, linked Program file or SL segment. It is at this stage that about half of the security implications of PM are realized. Obviously, this is a much higher level than that which the hardware imposes.

When the :PREP (or -ADDSL) command is entered, and the segment (or outer block, in the case of a program) is entered, the SEGMENTER checks two things. First, it checks to see if the proper capability is present. In the case of a program file, this implies that the user has entered CAP=PM on the :PREP command. The second check made, and one key to the implementation of MPE security regarding PM, is that the :PREP'ing user has PM capability assigned. If not, the :PREP (or -ADDSL) fails.

One exception must be noted at this point regarding the SEGMENTER's check of capabilities. In a program file, there are two different kind of segments. There is a special segment, called the OUTER BLOCK,

and there are all the other segments of the program. The Outer Block can be privileged in addition to the other segments. It is also possible for the other segments to be privileged and the Outer Block to not be, or vice versa. In any case, if the Outer Block is privileged, the :PREP will fail if CAP=PM is not specified. A minor point, but one worth mentioning.

The next level of capability checking is done when the program is actually loaded. In fact, the LOADER performs more checking than any other entity except perhaps the microcode. When the program is :RUN, the LOADER first checks to see if any segments are privileged. This is done by scanning the CST remapping array mentioned previously. If any segment is privileged, and the capability word stored in record 0 does not have the PM bit set on, then the load fails with "ILLEGAL CAPABILITY". The capability word is set by the SEGMENTER based upon the CAP= parm of the :PREP command. If none is specified, then IA,BA are assigned, but not if those capabilities do not exist at the group level.

Assuming that the capability word is in order, a check is made to be sure the capabilities of the program do not exceed that of the group. In other words, if IA,BA,PM are assigned to the program, all three must be present at the group level. (Also at the account level, but this not verified by the LOADER). It is important to note that the same checks are made for any privileged SL segments being loaded as a result of external calls made by the program. The only difference is that the LOADER differentiates between illegal capability of the program file and of the SL file in the error message. (Thank heavens for small favors!!!).

Compiler Options

We have seen so far what is necessary from the machine and operating system point of view to make a program privileged. Various areas were mentioned as having a PM bit set, such as the CST remapping array. However, we have not addressed how these bits magically get set. This section will deal with this subject. Since SPL is the only language that allows full access to PM code and machine instructions, this is the only compiler which will be shown.

As mentioned before, the smallest entity within the system that can be privileged is the segment. In SPL, the way a segment is given PM is by using the OPTION PRIVILEGED statement:

```
PROCEDURE EXAMPLE(A,B,C);
   VALUE A,B,C;
   INTEGER A,B,C;
   OPTION PRIVILEGED;
```

As mentioned before, IF ONE PROCEDURE IS DECLARED PRIVILEGED, THE WHOLE SEGMENT, AND ALL PROCEDURES WITHIN IT, ARE PRIVILEGED. It is therefore a good idea to place all privileged procedures together in one segment by using the $CONTROL SEG= compiler command. This applies equally to program file segments as well as segments that are compiled separately and placed in an SL.

To make a procedure UNCALLABLE, the following option is used:

```
PROCEDURE EXAMP;
  OPTION PRIVILEGED, UNCALLABLE;
```

This causes a bit in the Segment Transfer Table to be set. This table is resident at the end of every code segment, and is used by the PCAL instruction to branch to other segments. If this bit is on and current MODE bit is off, an STT VIOLATION occurs.

As previously mentioned, a program file has a special segment called an Outer Block. A special command is provided in SPL to make this segment privileged:

```
$CONTROL USLINIT,PRIVILEGED,MAIN=OB'
```

This option implies that the Outer Block is privileged from the start of the program, and remains privileged unless turned off. Since PCAL'ing a user mode procedure from a privileged one turns PM on, this would mean that the entire program will be privileged at all times. If this is not desired, then $CONTROL PRIVILEGED should be avoided.

:RUN Options, Intrinsics, and Things That Go BUMP in the Night

Several things need to be mentioned before any discussion of PM can be complete. First, besides the CAP = parm, there is another parameter of the :RUN command which deals with PM, and that is the NOPRIV option. NOPRIV is used by the LOADER to negate the effects of the PM bits in the prog file capability word and the CST Remapping Array. Thus, no part of the program will run in PM. However, if an SL segment is called, this

WILL run in PM. If this were not the case, then normal intrinsics, such as FOPEN, FREAD, etc, would not work. The implications are that if any PM code is executed, then a PRIVILEGED INSTRUCTION VIOLATION will occur. However, this is a good safeguard if development is being done and an added layer of security is necessary during testing.

There are several Intrinsics which behave differently when called from a PM segment. First, the file system will allow two things in PM. One is the ability to open Privileged files (files with a negative file code). This can only occur if the correct filecode is supplied along with being in PM. For filecodes equal or greater than zero, this is not necessary. The second ability granted by the file system is the ability to do nobuf, nowait IO to non-message files. (Message files can be accessed this way without PM).

The next group of procedures are the Data Segment Intrinsics. When called from user mode, this set of Intrinsics (GETDSEG, FREEDSEG, ALTDSEG, DMOVIN and DMOVOUT) checks for DS capability and returns a DST index, which is an index into a local table, not an MPE DST number. When called in PM, however, the check for DS capability is ignored, and the index returned is an actual MPE DST number. This implies that this number could then be used by MFDS, MTDS or MDS instructions later. Also, this must be done if SWITCHDB is to be called. SWITCHDB can only be called in privileged mode.

The GETPRIORITY Intrinsic has an option where a user can specify an absolute priority and place a process in a linear queue. Normally, the process will be placed in a circular queue. Obviously, this has far reaching ramifications, as a process in a high, linear queue could cause a lockout of other processes.

Probably the two most commonly used Intrinsics in the realm of PM are the GETPRIVMODE and GETUSERMODE instrinsics. These intrinsics allow a program :PREP'ed with CAP=PM to enter PM for a short time, then leave it. This is definitely the preferred method for performing privileged functions, especially if the amount of PM code needed is small and well contained. Typically, however, if the program is mostly privileged, it is probably better to just let it run in PM all the time. This allows the overhead of the calls to GETPRIVMODE and GETUSERMODE to be eliminated. Note also that using ;NOPRIV will cause these intrinsics to fail.

Deciding on PM

If you are a manager, you may still be uncertain whether or not to buy or use applications which run in Privileged Mode. The following are some guidelines.

If there is a real concern on your part, then discuss it with the vendor. Most vendors who use PM have not decided to do so lightly, as they realize that many people will have doubts and questions like you. Ask them if the PM was really necessary. Was it put in just to be a whiz bang, or is there some real benifit? Now that you've read this paper, you can put your knowledge to work. Is the PM just there to access some special Intrinsic capability, such as NOWAIT IO? If so, then it is probably very safe. Does the code look at MPE? If so, how secure is it? Is it looking at a part of MPE that can change drastically, or is it a part that has remained the same since the Series-CX? What is the reputation of the vendor? Are they noted for being MPE internals oriented, and knowing what is going on? Or did they pick someone off the street, teach him/her how to spell GETPRIVMODE and set them loose?

If you are really uncertain, sometimes you can get an independent consultant or HP involved. Hewlett-Packard SE's or System Specialists will sometimes be willing to look at an application and bless it "safe". Obviously, there is a large problem with liability. It will usually be up to the local management as to whether or not such a decision will be dealt out by one of their SE's. It is definately worth pursuing.

Finally, look at the vendor's support contract closely. If they guarantee that the code will work, then they must feel very good about it, and are confident that what they have written will not need much change, or at least that they will be able to change it if need be.

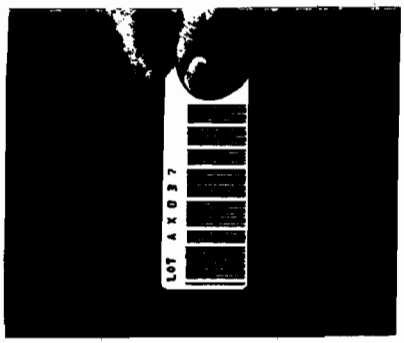
Conclusion

An attempt has been made to compile a compendium of options regarding Privileged Mode. In doing so, explanations were given so that it can be understood what is occuring during various privileged operations. It is hoped that these explanations would take some of the mystery of PM away, allowing a less fearful at- titude to form in the mind of the reader.

Name: Jason M. Goertz

Title: System Specialist Hewlett-Packard Neely Sales Region, Bellevue, WA. 98006

Jason started working with computers as a senior in high school. While attending Whitman College, he used computers in his course work. During his senior year, Whitman purchased an HP3000 Series II, which served as both an academic and administrative machine. Upon graduation, Jason stayed at Whitman as a Programmer/Data Base administrator, and later as system programmer, where he wrote a job scheduler, BEACON, and a security system, GUARDIAN, both of which employ Privileged Mode extensively. In addition, during this time, he worked on the HPIUG Contributed Library, and wrote or helped write many of the Infobase utilities for LIBS 4,5, 6 and 7. In 1980, Jason started working for the Bellevue HP office as a System Engineer, and within a year had added the role of Field Software Coordinator to his duties. He quickly became the area resource on MPE and internals, and has taught 4 MPE in- ternals classes, both to customers and to other HP employees. In August of 1983, he was promoted to System Specialist, where he concentrates on internals, special projects, and teaching of other SE's. Jason enjoys photography, backpacking and swimming for recreation, as well as travel and just relaxing. Jason's family consists of his wife, Doris, also involved in the HP com- munity, and his black cat, Nerd. All three reside in Redmond, Washington, north of Bellevue.



BAR CODES

ARE THEY FOR YOU ?

BRUCE M HENDERSON
INFORMATION RESOURCES
HEWLETT PACKARD
NOVEMBER 15, 1983

BAR CODES - ARE THEY FOR YOU?

THE USE OF BAR CODES IS BECOMING INCREASINGLY POPULAR IN MANY APPLICATION SYSTEMS AREAS. BUT WHEN DOES IT MAKE SENSE TO USE THEM, AND WHY? THE PURPOSE OF THIS PAPER IS TO EXAMINE:
THE ELEMENTS OF BAR CODES AND BAR CODE SYSTEMS,
WHAT THE ALTERNATIVES FOR HP3000 USERS ARE,
SEVERAL IMPLEMENTATIONS OF CURRENT BAR CODE TECHNOLOGY,
AND WHAT TO LOOK FOREWARD TO IN THE FUTURE.

BAR CODES ARE GRAPHIC REPRESENTATIONS, IN MACHINE READABLE FORMAT, OF CHARACTERS OF INFORMATION. THEY ALLOW RAPID DATA COLLECTION WITHOUT KEYING IT INTO A TERMINAL. THEY ASSIST IN SOLVING THREE MAJOR DATA COLLECTION PROBLEMS: THE COST OF CAPTURING DATA, THE TIMELINESS OF THE DATA CAPTURED AND THE ACCURACY OF THE DATA CAPTURED. THE WAYS IN WHICH BAR CODING SOLVES THESE DATA COLLECTION PROBLEMS WILL BE DETAILED THROUGHOUT THIS PAPER.

HISTORY

THE LARGEST SINGLE FACTOR IN ESTABLISHING BAR CODING AS A SUCCESSFUL NEW TECHNOLOGY HAS BEEN THE EXPERIENCE OF THE GROCERY INDUSTRY. WITH THE OBJECTIVE OF AUTOMATING SUPERMARKET CHECKOUT LINES, GROCERY RETAILERS AND SUPPLIERS AGREED TO COOPERATE. THE RESULT, ALMOST 80% OF ALL GROCERY STORE ITEMS ARE BAR CODED. SUPERMARKETS USING BAR CODE SCANNERS IN THEIR CHECKOUT COUNTERS ARE ABLE TO TAKE ADVANTAGE OF THIS, REALIZING IMPROVEMENTS IN INVENTORY CONTROL, MERCHANDIZING, AND SCHEDULING EMPLOYEES TIME.

THE AMERICAN BLOOD COMMISSION, ALONG WITH GROUPS FROM EUROPE AND AISA, HAVE ESTABLISHED THE USE OF BAR CODES IN A BLOOD CODING OPERATION. THIS HAS PROVEN VERY SUCCESSFUL FOR THOSE INVOLVED, WITH IMPROVED ACCURACY AND PRODUCTIVITY IN HANDLING BLOOD PRODUCTS.

A THIRD AND VERY IMPORTANT FACTOR IN BAR CODES, HAS BEEN THE ESTABLISHMENT OF LOGMARS. SET UP BY THE DEPT OF DEFENSE IN 1980, LOGMARS OBJECTIVES WERE TO ESTABLISH A MACHINE READABLE SYMBOLOGY TO BE MARKED BY COMMERICAL VENDORS
AND DOD ACTIVITIES ON ITEMS, UNIT PACKS, OUTER CONTAINERS AND SELECTED DOCUMENTATION, AND TO ESTABLISH PROCEDURES FOR USE OF THE SYMBOLOGY. THIS MEANT, ANYONE SUPPLYING THE DOD AFTER JULY 1982 HAD TO MOVE INTO BAR CODE TECHNOLOGY.

IN ADDITION TO THESE USES, THE BASIC NEED OF AMERICAN INDUSTRY TO COLLECT ACCURATE AND TIMELY DATA FROM AREAS INCLUDING MANUFACTURING, INSPECTION, TRANSPORTATION, AND INVENTORY HAS ADDED TO THE INCREASED POPULARITY OF BAR CODES

SYMBOLOGY

THE BASIC COMPONENT OF ANY BAR CODE IS TO REPRESENT ONE CHARACTER OF INFORMATION WITH A SYMBOL. THIS SYMBOL IS COMPOSED OF BLACK AND WHITE, NARROW AND WIDE LINES. THE LINES ARE REFERRED TO AS BARS. THE SPECIFIC ARRANGEMENT OF LINES (OR BARS) IS CALLED A BAR CODE SYMBOL FORMAT.

WIDE BARS AND WIDE SPACES ARE INTERPRETED AS BINARY 1. NARROW BARS AND NARROW SPACES ARE INTERPRETED AS BINARY 0. A BAR CODE READER (OR SCANNER) REALLY ONLY LOOKS AT THE CHARACTERISTICS OF BLACK OR WHITE, AND WIDE OR NARROW.

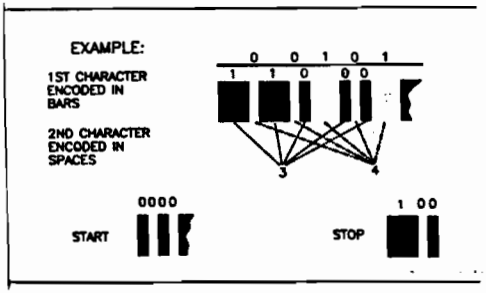
QUIET ZONES MUST BE PROVIDED AT BOTH ENDS OF THE BAR CODE. THEY PROVIDE SEPARATION OF THE PRINTED AREA FROM THE ACTUAL BAR CODE. AT THE LEFT (OR PRECEEDING THE FIRST CHARACTER) OF THE BAR CODE, IS THE START CODE. THE STOP CODE IS LOCATED AT THE RIGHT OF THE BAR CODE. START AND STOP CHARACTERS ARE USED BY THE SCANNER TO IDENTIFY THE ACTUAL BEGINNING AND ENDING OF EACH BAR CODE. THEY ALSO PERMIT BIDIRECTIONAL READING.

OPTIONAL CHECKSUM CHARACTERS MAY ALSO BE INCLUDED IN THE BAR CODE TO IMPROVE RELIABILITY. THE VALUE OF THE CHECKSUM IS DETERMINED BY AND ALGORITHM PERFORMED ON THE BAR CODE DATA CHARACTERS. A CHECKSUM IF USED, BECOMES THE LAST CHARACTER OF THE SYMBOL (PRECEEDING THE STOP CHARACTER).

OVER 25 DIFFERENT SYMBOLOGIES HAVE BEEN DEVELOPED OVER THE LAST 15 YEARS. WE WILL EXAMINE TWO OF THE MOST POPULAR INDUSTRIAL SYMBOLOGIES.

INTERLEAVED TWO OF FIVE

INTERLEAVED TWO OF FIVE IS A SELF CHECKING CONTINUOUS CODE USED TO REPESENT THE CHARACTERS 0-9. IT HAS BEEN WIDELY ACCEPTED IN WAREHOUSING AND HEAVY INDUSTRY. THE DISTRIBUTION SYMBOLOGY STUDY GROUP HAS RECOMMENDED INTERLEAVED 2 OF 5 AS A STANDARD FOR NUMERIC LABELING OF CORRIGATED SHIPPING CONTAINERS. BECAUSE OF ITS HIGH DENSITY (SMALLER BAR CODES) IT HAS BEEN GAINING POPULARITY. HOWEVER, MANY APPLICAIONS REQUIRE THE CODING OF ALPHANUMERIC DATA, WHICH 2 OF 5 WILL NOT DO.



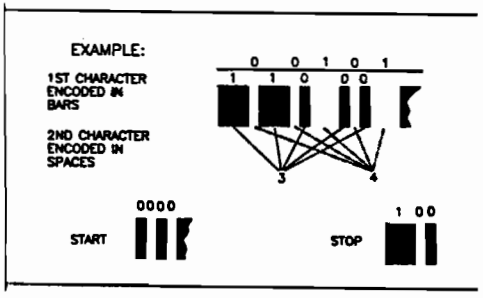
3 OF 9.

3 OF 9, ALSO KNOWN AS CODE 39, IS BY FAR THE MOST POPULAR ALPHA NUMERIC SYMBOLOGY. IT HAS BEEN ACCEPTED AS THE STANDARD BY BOTH LOGMARS AND THE DISTRIBUTION SYMBOLOGY STUDY GROUP. 3 OF 9 IS A 9 ELEMENT CODE, WITH 3 ELEMENTS BEING WIDE AND 6 BEING NARROW (HENCE ITS NAME). BOTH BAR AND SPACES ARE POSITIONED TO ENCODE LOGIC VALUES. 43 ALPHANUMERIC CHARACTERS, PLUS START AND STOP ARE ENCODABLE. WHEN COMPARED WITH INTERLEAVED 2 OF 5 CODE 39 IS TWICE AS LONG FOR THE SAME NUMBER OF CHARACTERS, BUT DOES OF COURSE HAVE ALPHANUMERIC CODING CAPABILITIES.

---

3 of 9 Code Character Set Encodation

CHAR.	PATTERN	BARS	SPACES	CHAR.	PATTERN	BARS	SPACES
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	-		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	SPACE		01010	1000
H		10010	0010	$		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		01001	0001	%		00000	0111



BAR CODE SYSTEMS

A BAR CODE SYSTEM CONSISTS, MOST BASICALLY OF STORING AND RETRIEVING DATA. IT IS POSSIBLE HOWEVER, TO BREAK DOWN THESE COMPONENTS INTO A MORE MEANINGFUL GROUPING.

1. What data will be stored in bar code format? Singular pieces of information used for key identification purposes are good candidates. If this information is entered on a repetitive basis, or accuracy is a critical priority, bar codes should be considered. Examples of this type of data are: item or part numbers in an inventory control system, item numbers and location numbers in a warehouse tracking system, work order numbers in work in process tracking system, and purchase order numbers in a material receiving system. It is also becomming increasingly popular to have a host computer generate the identification keys for bar coding in different tracking systems. An example would be one Move Ticket number generated to track a specific item number received on a specific purchase order.

2. How will the data be stored?
Storage in this case means the physical bar coded data. It may be a paper document, a label, etc..
What symbology will be used? Typically if alphanumeric data must be encoded, code 39 will be the choice. If only numeric data is required the choices are numerous.
What media will be used to hold the bar coded data? Plastic coated labels may be required in some environments. Maybe your normal printer paper stock is fine. How this media will be physically utilized in the operation is one of the main determining factors.
How the bar code data will be printed is of major importance. Print size, ink quality, and print technique all impact the readability of a bar code.
And what good is a bar code if you can't read it?

3. How will the data be retrieved (or read) ? Encoded data must be read by an optical scanner that in turn sends a logic signal to a decoder. The decoder then translates the scanned data into computer readable form. Bar code readers must be capable of accurately handling this operation. The type of reader selected is determined by physical layout, system flow, the need for real time updates, and other system hardware.

4. How is a successful system measured? Determining what is to be read, how it will be stored, and how to retrieve it requires careful planning, selection, and implementation. As with any complex integration problem, success is not guaranteed.
One method of measuring success is to examine what is called the first read rate. Since accurate, fast, and cost effective data capture is the underlying objective in using bar codes, the number of scans required to read any one bar code is critical. A first read rate (number of times a bar code is read on the first scan) of 80% should be considered the minimum for success. It is important to note that if a bar code is not read on the first scan, it has not been misread. It must simply be re-scanned until it is successfully read. Actual misreading of a bar code is called error substitution.
The error substitution rate (expressed in characters) should be better than one in a million when using code 39.

BAR CODE HARDWARE

PRINTING BAR CODES

There are two basic considerations in printing: what to print on (the media) and what to print with. With media, readability is of course a primary factor, however durability is equally important. If time and environment will damage a bar code, it's readability is gone. In selecting a media, look first at its optical characteristics, then at how those characteristics can be preserved (possibly plastic coating). It is not uncommon to mix media in a bar code system. Using adhesive label's on storage bins with printed material move documents, for example, combines two different technologies. Make certain that both produce compatible bar codes, ie. can be scanned

The choice of printers breaks out into three catagories: volume - how many labels or documents are required in a particular time frame, resolution - is the density and print quality good, and cost - is the printer cost effective for the application.

TYPE	VOLUME	RESOLUTION	COST
Formed Impact - Drummer	Low/Medium	High	Low/Medium
Formed Impact - Daisywheel	Low/Medium	Low/Medium	Low/Medium
Electrostatic	High	Medium	High
Laser	High	Low/Medium	High
Ink Jet	High	Low/Medium	High
Thermal	Low	Medium	Low
Dot Matrix	Low	Low	Low

There is also another choice, not listed above. That is, to have bar codes printed commercially. In a situation where bar codes (usually labels) may be pre-assigned and therefore pre-printed, commercial printing should always be considered. Where pre-assignment is not possible because the content of the bar code is not known long before printing, then using an in house printer for "on-demand" bar codes is the solution.

Physical layout for on-demand printing usually means having a printer located near the place where the bar code will be needed. This translates into multiple printers and generally rules out the high cost choices. Once a cost range is determined, volume must be considered. On demand printing does not always imply low volume. Placing one slow printer, like a thermal, in a critical path operation will undoubtably create a bottleneck in the system flow. Resolution is important not only for readability, but for bar code size. Many labels, and certain documents, leave available space for only very small bar codes.

Printer flexibility is another important area to consider. Dot matrix tends to be the most flexible of the low cost printers, being able to produce documents and labels. The drummer printer, which has the best resolution, can only print labels. Thermal printers lack media flexibility.

For the HP3000 user, printer choices are fairly broad. Hewlett Packard manufactures an excellent high end laser printer - the 2680. Using the 2680 to print bar codes would satisfy anyones need for real high volume output, making even commercial printing unnecessary. Where low/moderate volumes of labels or documents are required, the HP2631G (Option 200) dot matrix printer will do the work in a very cost effective way. Other manufacturers who have products in the lower cost end of the market are Intermec and Scanmark. They have specialized in the drummer type printers, used to print bar code labels.

READING BAR CODES.

Scanners for reading bar codes can be split into two catagories: fixed and hand held. Fixed scanners (or moving beam scanners) are generally laser and read the bar codes as they move past the scanner. Automated material handling conveyors are one example. Bar coded containers move on a conveyor past a fixed scanner.
As the scanner interprets the code, it is sent automatically to the decoder and then to an on-line computer. Automated conveyors with fixed head scanners like these, can be used for product sortation after order picking.

Hand held scanners are the type where a wand, or sometimes a gun, is passed over the stationary bar code. With a wand, the operator places it in the quiet zone, making light contact with the code surface. The wand is then pulled across the entire code, as if drawing a light pencil line. Normally the bar code reader will emit a beep if the code was read correctly. This operation works very well on flat surfaces, but not so well on curved surfaces. The height of the bar is also an important factor. There is a natural tendency to move the wand through the top or bottom of the code before scanning the entire thing. To minimize this, all bars should be at least 0.3 inches in height. Where the entire code is excessively long, even 0.3 inches may not be sufficient.

Hand held scanners can be either on line, or portable. A typical on line application example would involve attaching the bar code reader to an on line terminal. As PO's are received, the hand held wand would be used to scan a bar coded PO number. When this data is sent to the host computer, it will look (to the computer) as if the data came from the terminal.

A portable bar code reader typically collects data (scanned or keyed) in its memory until the collection process is done. The reader is then taken to a download station where, by using a direct cable to the computer, all the data stored in the portable can be downloaded to a host CPU. An example of this would be stockroom cycle counting. The operator would go to the stockroom with a portable reader and a cycle count report. The item number from the report would be scanned, the bar coded loction on the bin would be scanned, and the quantity counted keyed in. Once all counts were complete the bar code reader would be taken to a work station for downloading.

Currently HP has four offerings in the hand held scanner catagory. The HP3075A and HP3076A are complete data capture terminals designed for the manufacturing environment. These terminals can be used with VPLUS and data capture on the HP3000 and HP1000. They will read bar codes, badges, and include a small CRT display. Each terminal occupies a seperate port and must be configured in to the HP3000 just like any other terminal.

The 16800A is a dedicated bar code reader with an RS232 port that will attach to any HP terminal. One very powerful feature of this device is that it actually shares a line and port with an existing terminal. This is called evesdrop mode, and means the HP3000 will communicate with the 16800A and a terminal at the same time, on the same port. Data entry can be done using THE 16800A ALONE, THE TERMINAL ALONE, OR READING BAR CODES WITH THE 16800A AND INPUTING ADDITIONAL DATA FROM THE TERMINAL.
WITH EXISTING VPLUS APPLICTIONS
THE 16800A CAN BE CABLED IN AND USED TO READ BAR CODES WITH ZERO PROGRAMMING CHANGES! IT IS ALSO PROGRAMMABLE.

The HP92911A is a similar concept to the 16800A. It is a less expensive dedicated bar code reader that attaches to any HP262X terminal. It tpo requires no special software or reprogramming, but does require that ENTER or RETURN be pressed to transmit the result of each bar code read to the computer.

Other manufacturers supplying bar code readers include INTERMEC, MSI, and Telxon. All three manufacture portable bar code readers which are compatible with the HP3000, in varying degrees.

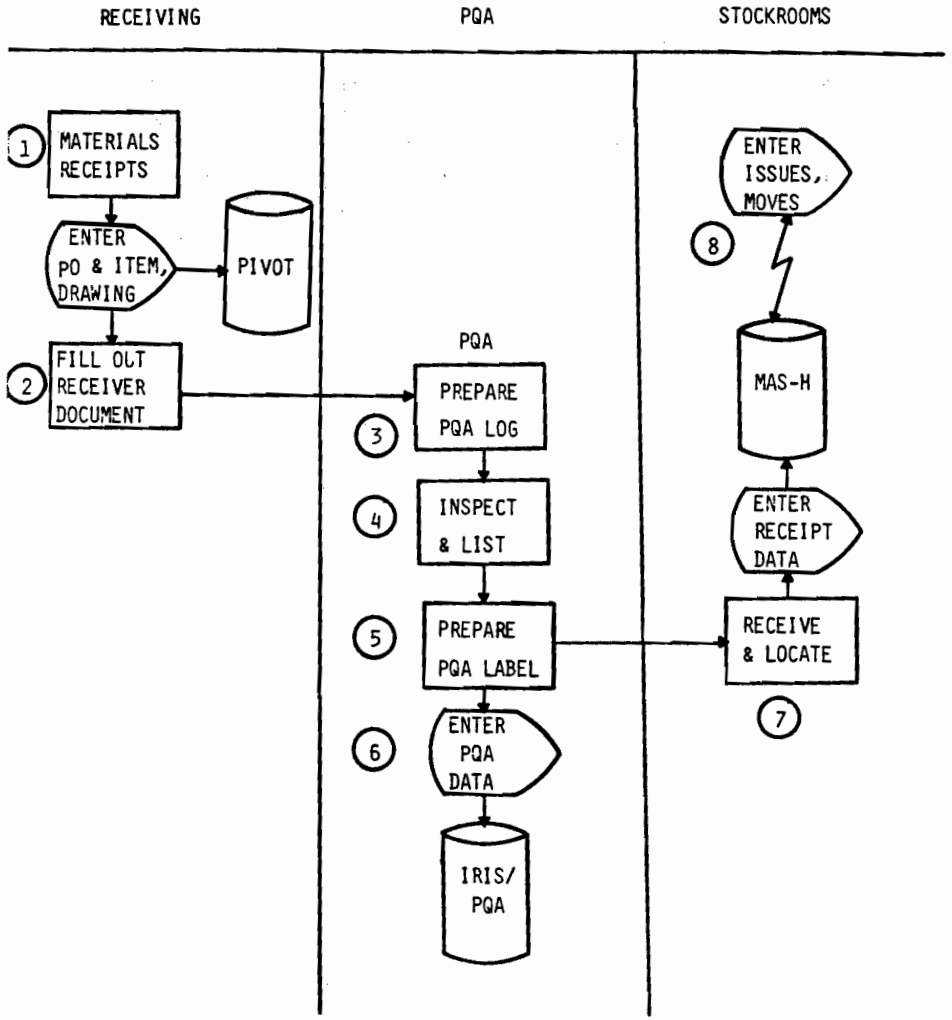
APPLICATION EXAMPLES

MANUFACTURING

General Electrics Nuclear Control and Instrumentation Division in San Jose , Calif. processes over 5000 inventory transactions a day through its manufacturing system. This manufacturing system is a hybrid combonaion of Martin Marietta's MASH package and several locally developed systems (PIVOT & IRIS). Originally running on two seperate HP3000's with no realtime interface the flow went in to receiving (PIVOT), then to PQA - inspection (IRIS/PQA), and then to the stockroom (MASH). Once material was received into the stockroom it could be issued or moved using MASH. All this created the need for redundant data entry, timing problems, and accompanying lack of accuracy.
See the "PRESENT METHODOLOGY" flow.

AUTOMATED DATA CAPTURE

PRESENT METHODOLOGY



It was decided that two courses of action needed to be taken to help minimize these problems: the use of automated data capture and creation of a common integrated material tracking data base.
See the "Project Scope".

AUTOMATED DATA CAPTURE

PROJECT SCOPE

PURPOSE: INCREASE INVENTORY SYSTEM ACCURACY WHILE REDUCING MANUAL DATA ENTRY REQUIREMENTS

OBJECTIVES:

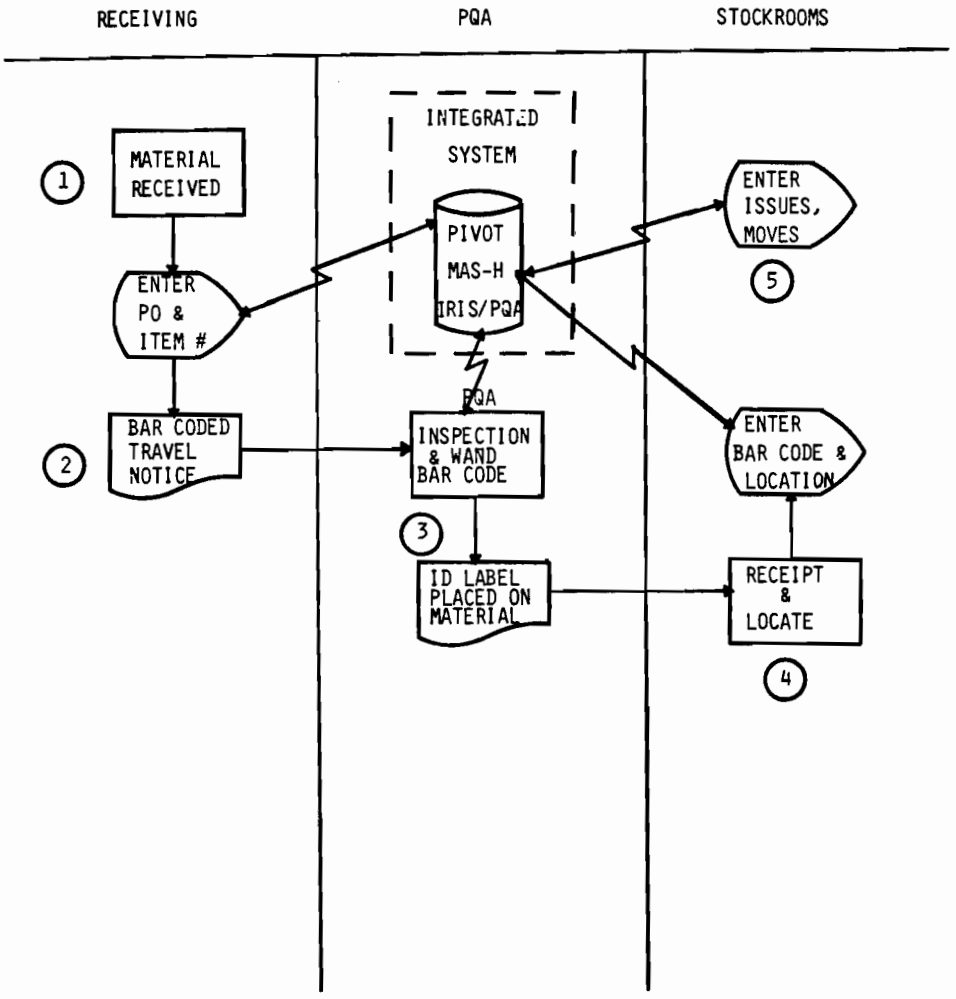
- REAL TIME DATA CAPTURE
  - ELIMINATE TIMING PROBLEM
- BAR-CODE DATA ENTRY
  - ELIMINATE MANUAL DATA ENTRY ERRORS
  - AUTOMATIC ENTRY OF MATERIAL LOCATIONS
  - INCREASES SPEED AND ACCURACY OF INVENTORY COUNTING PROCESS
- TRACKING MATERIAL
  - COMMON DATA BASE
  - ONE COMPUTER SYSTEM

A new system flow, utilizing bar codes for data capture, was identified (see BAR CODE MATERIAL TRACKING PLANNED METHODOLOGY).

AUTOMATED DATA CAPTURE

BAR-CODE MATERIAL TRACKING

PLANNED METHODOLOGY



Under this flow five steps were identified:

1. Material Received.
   PO Number and Item-no are entered into a receiving screen.
   The receipt is recorded and added to a common tracking data base.
2. A Travel Notice (like a move ticket) with a bar coded Travel Notice number, is printed at the receiving work station. One travel notice for each individual PO/ITEM-NO combinaton is printed.
3. As material arrives at inspection (P0A) the bar coded TN number is scanned real time to change it's locaion. Once inspection is complete, bar coded labels for all parts are printed and applied individually to each part as it is packaged.
4. Once packaged and labeled, material leaves P0A and arrives at the stockroom. The TN number is scanned to update its location. A material handler takes the parts and puts them away. Using a portable hand held reader, the item-no and store location are scanned and the putaway quantity is keyed in.
5. Material can be moved and issued by using a portable reader at the store location: scan item-no and store location, key in work order-no, and key in quantity. Eventually work order-no will also be bar coded. Optionally a real-time hand held device can be used to record those transactions by using the accompanying paperwork.

Looking at this system strictly from the bar code perspective, four different operations (and types of hardware) are being used:

1. Printing bar coded documents - the TN document.
2. Scanning the TN number with a hand held real time reader.
3. Printing bar code labels, one for each part in a high volume.
4. Scanning the bar coded label and storage location label with a portable hand held device. Key quantity into this same device. When the portable reader is full, download the data to the HP3000.

Additional operational restrictions were the capability to print a bar code label in a minimum of four seconds, being able to print a bar code label that could represent 22 code39 characters in less than 3 inches of width, and making sure all hardware was fully compatible with HP3000. Since both item-no and location were alpha-numeric characters, code39 was chosen. All bar code hardware had to be able to work with code 39.

For printing the bar coded TN document the HP2631G, option 200, was selected. Since its only IO port is an HPIB, it must be driven from an HP2648A graphics terminal on the HP3000. Although the option 200 printer has the capability to produce bar codes, there is no software package available with it to do this. Consequently it was necessary to write a COBOL program sending different series of escape sequences through the HP2648A to the HP2631G. Fortunately the instructions for doing this are documented fairly will in the option 200 manual.

The HP16800A was selected as the real time hand held reader. This device was so easy to use and integrate into the existing system, that 10 minutes after unpacking it, the reader was fully functional. The only setup required is checking and possibly resetting 24 toggle switches on the back of the unit.

Hardware selection for printing bar code labels was more difficult. The bar code size and print speed requirements ruled out dot matrix. Cost constraints eliminated the high cost printers. Finally a drummer type printer manufactured by INTERMEC(8220) was selected. In spite of the fact that this printer has an RS232 port and should communicate directly with the HP3000, it lacks the necessary internal protocol. This was solved by using a data concentrator to dirve all the label printer. This device is basically a 16 port multiplexor, with special firmware for an HP3000 interface.

However no HP3000 foreign device interface is that easy. The INTERMEC data concentrator will only work in an ENQ/ACK protocol. This means the data concentrator must be driven by a special SPL program that totally controls its protocol.

Selection of a portable hand held device was based as much on the ability to interface to the HP3000 as the ability to collect the required data. The hand held portable had to be programmable for structuring the input data, plus easy for the operators to learn and use. All the data collected in the portable had to be directly downloadable to the HP3000 via an RS232 line.

Although several portables would handle the data collection operation, the Telxon was judged to be easiest to interface with the HP3000. Working on a DC1/DC2 protocol, data can be copied from the Telxon terminal to the HP3000 as simply as using FCOPY. A specific application program was written however, to handle the communications and error routines.

For a major system implementation, the the bar code system was installed with very few problems and, once they were corrected, was very well received by the users. The major problems encountered during implementation were:

1. Having to rework the basic data concentrator protocol. This was due to innadequate testing with multiple system users printing bar codes.
2. Having an initial first read rate below 50%. Printing bar code labels that were too long for their height caused this problem. For a 22 character item-no printed using code 39, over 2 1/2 inches in length were required while the height remained 0.3 inches. With those particular dimensions it is almost impossible for even the steadiest of hands to successfully scan the bar coded label. The solution was to remove lot-no from the right most portion of item-no effectively reducing it to 16 characters in length. This in turn allowed the first read rate to climb to well over 80%.
3. During the first few days of using portables, the material handler would not turn them in for downloading until the terminal's memory had filled up. In two cases that meant loosing some the data that had been collected. It also made the data less real time. The solution was for every material handler to turn in each terminal at least every couple hours. That way the data is close to real time, and the terminal memories are impossible to over fill.

No other major problems were encountered upon implementation. This wa due in a large part, to almost totally planning out a good bar code system strategy. Making sure that a bar code created on one type of printer can be interpreted by both types of readers. The volume levels and size requirement were planned out in the beginning. The system is capable of producing the right types of bar codes at the time they're needed. And the data elements chosen to be bar coded were the right ones. They provide for optimal cost effectiveness and the efficiency of saving time.

DISTRIBUTION

The area of distribution, with its critical high volume material flows and its need for fast turnaround provides an excellent area for further bar code application. One such opportunity is within the new Warehouse Management (WM3000) package, currently being developed ar HP. This project is somewhat unique in that the use of bar codes is being integrated into the system design.

The front end warehouse flow goes from receiving, through any inspection or packaging areas, to putaway. Putaway includes producing a selected (on demand) routed putaway report. The other half of the system flow starts with picking, goes through packaging, to shipping. Bar code applications will be implemented at the following points:

1. Receiving.
   As a single purchase order line item is received, a bar coded move document can be automatically printed. If an entire truck load shipment is to be received at once, a bar coded report listing everything on the shipment can be printed.
2. Tracking
   As each receipt moves through different areas of the warehouse its path can be recorded and tracked by, minimally, scanning the move document number with a real time hand held device.
3. Putaway.
   Rather than printing a requested putaway report, it can be optionally downloaded to a portable hand held reader. The material handler then takes only the portable reader and the items to be putaway out to the stock area. In serial fashion each location will appear directing the person where to go. As each item is putaway, the item-no is scanned and the putway quantity verified. Then the next location to be putaway is displayed. When the putaway is complete, the portable reader is returned to a work station where the putaway confirmation data is downloaded to the HP3000.
4. Cycle Counts.
   Cycle counts can work one of two ways, using a printed cycle count report or downloading the proposed count information to a portable hand held device (very similar to putaway). The portable device, if used, would direct the material handler throughout the warehouse. When all cycle counts are complete, the portable is returned to a work station and its data downloaded to the HP3000.
5. Picking.
   Based upon a variety of criteria a pick list is requested. It can be printed, or loaded to a portable bar code reader. From here on in, the scenario is like putaway going in reverse.
6. Packaging/Shipping.
   As each line item passes through the packaging and shipping area, the bar coded pick number is scanned to automatically record its movement.

To ensure optimal integration and ease of use, the entire bar code system will use only HP hardware, including some new low cost bar coding equipement.

THE FUTURE

Before discussing the specific future of bar codes, what about other possible alternatives?

OCR - Optical Character Recognition.

OCR uses a special style of human readable characters which can be read by page readers and hand held wands. Hand held wands have not caught on because of expense and limitations (they can only read 0-9 plus 8 alpha characters). The number of misreads today is significantly higher than for bar codes. Until low reader cost and virtually error free reading can be offerred, OCR will gain little in popularity.

Magnetic Stripe.

Magnetic Stripe is prohibitively expensive both to print and read. Until some breakthru in cost occurs, it will remain beyond the reach of most systems.

Bar codes on the other hand, are getting cheaper, more popular, and seem to be on the steeply rising portion of the life cycle curve. Things to look for in the near future are less expensive, more versitile products. HP is making a comitment in this area.

There are already a lot of very good, very advanced technologies available today. The use of hand held laser scanners will continue to grow. Although superior to wand scanners, they currently cost a lot more - enough to negate most of the relative benefit.

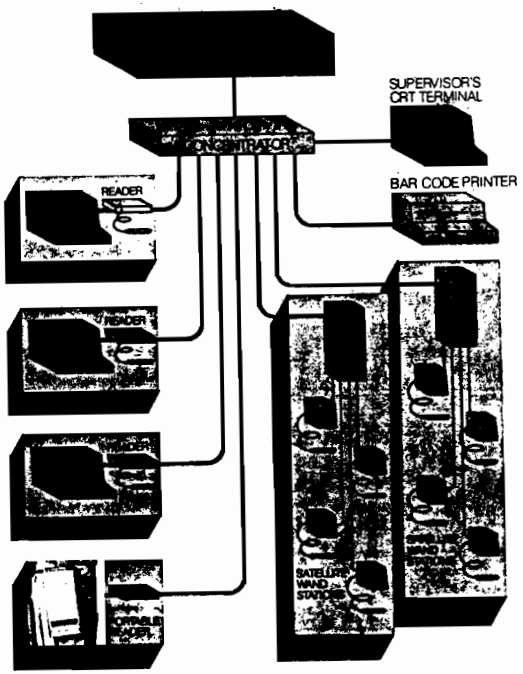
Automated material handling equipment is here, but must become more cost effective. Look for software packages like WM3000 to commonly control product sortation systems in the near future. And soon after, the control of stacker cranes and forklifts.

CONCLUSION

When trying to design a bar code system, first make sure your problem is one of data collection. Is there a problem with data costs, accurracy, and timeliness? If you're convinced that bar codes are the solution to your problem, then approach their use very systematically:

1. What data element should be bar coded?
2. How should the data be stored?
3. How should it be retrieved?
4. Set up a plan for measuring the success or failure of your bar code implementation.

And finally, when your ready to make a decision: select good quality equipment, make sure it will interface well with the HP3000, and ensure it can be serviced quickly and effectively.



Use of COBOL II Macros for Database Access

Jerrel Baxter, CDP
School of Optometry

INTRODUCTION

IMAGE/3000 provides a powerful tool for development of data storage and retrieval applications. Its use of subroutine CALL's for operation invocation allows fairly simple inclusion into all programming languages for the Hewlett Packard 3000. However, this CALL level interface also places total responsibility for error checking and reporting on the programmer which can require rather voluminous coding. This paper describes one method for dealing with this problem for COBOL II programmers, use of COBOL II macros. The methods used are also applicable to use of VPLUS/3000 for terminal screen handling and DSG/3000 for graphics generation.

STANDARDS

In order to create an orderly process for generating code for database references the following standards were adopted.

1. A COPYLIB module defining a parameter set for use in all programs with database references was created. See Appendix A for a listing of the defined module.

2. Naming conventions for creating reference variables for database names, dataset names, search item names, search item values, and dataset record layouts were defined and made mandatory for all programs.

3. COBOL II macros were defined for making all database calls. Use of the COBOL II macros guarantees adherence to standard error checking and makes the purpose of many database calls simpler to recognize. The use of the macros also makes it easy to incorporate enhancements to the error notification function in the future by replacing the current error handling code within the macro definitions.

As an example the following shows DATA DIVISION and PROCEDURE DIVISION code segments for a simple database referencing program:

```
:FILE COPYLIB=COPYLIB.COBOL
:FILE COPYDEF=COPYDEF.COBOL
 .
 .
 .
$CONTROL USLINIT,NOMIXED,NOLIST
$INCLUDE COPYDEF.COBOL
 .
 .
 .
DATA DIVISION.
WORKING-STORAGE SECITON.
COPY DBPARAMS.
01  DB-CLIENT                  PIC X(10) VALUE "  CLIENT  ".
```

```
       01  DS-M-IDENTIFICATION      PIC X(16) VALUE "M-IDENTIFICATION".
       01  DS-D-PROJECTS            PIC X(16) VALUE "D-PROJECTS      ".
       01  DS-A-SHORTNAMES          PIC X(16) VALUE "A-SHORTNAMES    ".

      (Continued on next page.)
       01  DI-ACCOUNT               PIC X(16) VALUE "ACCOUNT         ".
       01  DV-ACCOUNT               PIC 9(4) DISPLAY.
       01  DI-SHORTNAME             PIC X(16) VALUE "SHORTNAME       ".
       01  DV-SHORTNAME             PIC X(6).
       01  DR-M-IDENTIFICATION.
       COPY MIDENTIF.
       01  DR-D-PROJECTS.
       COPY DPROJECT.
       01  DR-A-SHORTNAMES.
       COPY ASHORTNA.
        .
        .
        .
       PROCEDURE DIVISION.
       MAIN-PROCESSING SECTION.
       INITIALIZATION.
           %DBOPEN(DB-CLIENT#,1#)
       CONTINUE-PROCESSING.
        .
        .
        .
      * Note use of V version of COBOL II macros to indicate VPLUS/3000
      * is active (error routine closes VPLUS form and terminal files
      * prior to aborting).
           %VDBFIND(DB-CLIENT#,DS-D-PROJECTS#,DI-SHORTNAME#
                   ,DV-SHORTNAME#)
           IF DB-MISSING
             MOVE "Client not found." TO V-MESSAGE
             GO TO CONTINUE-PROCESSING.
           %VDBGET(DB-CLIENT#,DS-D-PROJECTS#,5#,DB-ALL-LIST#
                   ,DR-D-PROJECTS#,DB-DUMMY#)
           IF DB-EOC
             MOVE "Client not found." TO V-MESSAGE
             GO TO CONTINUE-PROCESSING.
        .
        .
        .
      * End of program.
```

DEFINITIONS

The standard macro definitions and an example of their calling sequence are listed below:

```
      $DEFINE %DBOPEN=
      * %DBOPEN(DB-BASENAME#,1#)
      * Opens the specified database in the specified mode.  On open
      * failure an error message is output, the data base is closed
    * (per IMAGE manual recommendations), and the program is aborted.
       CALL "DBOPEN" USING !1
          DB-PASSWORD
          DB-MODE-!2
          DB-STATUS
       IF NOT DB-OKAY
         CALL "DBEXPLAIN" USING DB-STATUS
         CALL "DBCLOSE" USING !1
           DB-DUMMY

    (Continued on next page.)
           DB-MODE-1
```

```
          DB-STATUS
        STOP RUN.#

$DEFINE %DBCLOSE=
* %DBCLOSE(DB-BASENAME#)
* Closes the specified data base. On close failure an error
* message is output and the program is aborted.
    CALL "DBCLOSE" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBREWIND=
* %DBREWIND(DB-BASENAME#,DS-M-SETNAME#)
* Rewinds the specified dataset within the specified database.
* On rewind failure an error message is output and the program
* is aborted.
    CALL "DBCLOSE" USING !1
        !2
        DB-MODE-3
        DB-STATUS
    IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBBEGIN=
* %DBBEGIN(DB-BASENAME#)
* Outputs a text message to the IMAGE logging file and marks the
* beginning of a logical transaction. On failure an error message
* is output and the program is aborted.
    CALL "DBBEGIN" USING !1
        DB-TEXT
        DB-MODE-1
        DB-STATUS
        DB-TEXT-LENGTH
    IF NOT (DB-OKAY OR DB-LOGGING-DISABLED)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBEND=
* %DBEND(DB-BASENAME#)
* Outputs a text message to IMAGE logging file and marks the end of
* a logical transaction. On failure an error message is output and
* the program is aborted.
    CALL "DBEND" USING !1
        DB-TEXT
        DB-MODE-1
        DB-STATUS
        DB-TEXT-LENGTH
    IF NOT (DB-OKAY OR DB-LOGGING-DISABLED)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#
$DEFINE %DBDELETE=
* %DBDELETE(DB-BASENAME#,DS-M-SETNAME#)
* Deletes the currently referenced data record in the specified
* dataset for the specified database. On failure an error
* message is output and the program is aborted. A missing record
* or chain head is not considered to be a failure.
    CALL "DBDELETE" USING !1
        !2
        DB-MODE-1
        DB-STATUS
    IF NOT (DB-OKAY OR DB-MISSING OR DB-CHAIN-HEAD)
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#
```

```
$DEFINE %DBFIND=
* %DBFIND(DB-BASENAME#,DS-D-SETNAME#,DI-KEYNAME#,DV-KEY-NAME#)
* Finds the search path for the specified key value for the
* specified key item in the specified dataset and database. On
* failure an error message is output and the program is aborted.
* A missing key value is not considered to be a failure.
    CALL "DBFIND" USING !1
       !2
       DB-MODE-1
       DB-STATUS
       !3
       !4
    IF NOT (DB-OKAY OR DB-MISSING)
     CALL "DBEXPLAIN" USING DB-STATUS
     STOP RUN.#

$DEFINE %DBGET=
* %DBGET(DB-BASENAME#,DS-M-SETNAME#,7#,DB-ALL-LIST#
*       ,DR-M-SETNAME#,DV-KEYNAME#)
* Gets the data record corresponding to the parameter values. On
* failure an error message is output and the program is aborted.
* Beginning of file, end of file, beginning of chain, end of chain,
* and no entry are not considered failures.
    CALL "DBGET" USING !1
       !2
       DB-MODE-!3
       DB-STATUS
       !4
       !5
       !6
    IF NOT DB-GET-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     STOP RUN.#

$DEFINE %DBLOCKBASE=
* %DBLOCKBASE(DB-BASENAME#)
* Locks the specified database. On failure guarantees release of
* all locks, outputs an error message and aborts the program.
    CALL "DBLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
```

(Continued on next page.)

```
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
      STOP RUN
    ELSE
      IF NOT DB-OKAY
        CALL "DBEXPLAIN" USING DB-STATUS
        STOP RUN.#

$DEFINE %DBLOCKSET=
* %DBLOCKSET(DB-BASENAME#,DS-M-SETNAME#)
* Locks the specified dataset for the specified database. On
* failure guarantees release of all locks, outputs an error message
* and aborts the program.
    CALL "DBLOCK" USING !1
       !2
       DB-MODE-3
       DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
```

```
      CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
      STOP RUN
     ELSE
      IF NOT DB-OKAY
       CALL "DBEXPLAIN" USING DB-STATUS
       STOP RUN.#

$DEFINE %DBLOCKENTRY=
* %DBLOCKENTRY(DB-BASENAME#,DE-LOCKSPECIFICATION#)
* Locks according to the lock specifications in the specified
* table. On failure guarantees release of all locks, outputs an
* error message and aborts the program.
    CALL "DBLOCK" USING !1
      !2
      DB-MODE-5
      DB-STATUS
    IF DB-CONDITION > 0
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
     STOP RUN
    ELSE
     IF NOT DB-OKAY
      CALL "DBEXPLAIN" USING DB-STATUS
      STOP RUN.#

$DEFINE %DBPUTALL=
* %DBPUTALL(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Puts the data record to the specified dataset and database using
* @; for the record list. On failure generates an error message,
* unlocks the database (assumes normal use of mode 1 for DBOPEN)
* and aborts the program.

(Continued on next page.)
    CALL "DBPUT" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      DB-ALL-LIST
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
     STOP RUN.#

$DEFINE %DBPUTSAME=
* %DBPUTSAME(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Puts the data record to the specified dataset and database using
* *; for the record list. On failure generates an error message,
* unlocks the database (assumes normal use of mode 1 for the
* DBOPEN), and aborts the program.
    CALL "DBPUT" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      DB-SAME-LIST
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
```

```
      CALL "DBUNLOCK" USING !1
         DB-DUMMY
         DB-MODE-1
         DB-STATUS
      STOP RUN.#

$DEFINE %DBPUT=
* %DBPUT(DB-BASENAME#,DS-M-SETNAME#,DA-M-SETNAME#
*       ,DL-ITEMLIST#)
* Puts the specified record to the specified dataset and database
* using an abbreviated item list and record area.  On failure
* outputs an error message, unlocks the database (assumes normal
* use of mode 1 for DBOPEN) and aborts the program.
   CALL "DBPUT" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      !4
      !3
   IF NOT DB-OKAY
    CALL "DBEXPLAIN" USING DB-STATUS
    CALL "DBUNLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    STOP RUN.#
$DEFINE %DBUNLOCK=
* %DBUNLOCK(DB-BASENAME#)
* Unlocks all held locks for the specified database.  On failure
* outputs an error message and aborts the program.
   CALL "DBUNLOCK" USING !1
      DB-DUMMY
      DB-MODE-1
      DB-STATUS
   IF NOT DB-OKAY
    CALL "DBEXPLAIN" USING DB-STATUS
    STOP RUN.#

$DEFINE %DBUPDATEALL=
* %DBUPDATEALL(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Updates the data record to the specified dataset and database
* using @; for the record list.  On failure generates an error
* message, unlocks the database (assumes normal use of mode 1 for
* DBOPEN) and aborts the program.
   CALL "DBUPDATE" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      DB-ALL-LIST
      !3
   IF NOT DB-OKAY
    CALL "DBEXPLAIN" USING DB-STATUS
    CALL "DBUNLOCK" USING !1
       DB-DUMMY
       DB-MODE-1
       DB-STATUS
    STOP RUN.#

$DEFINE %DBUPDATESAME=
* %DBUPDATESAME(DB-BASENAME#,DS-M-SETNAME#,DR-M-SETNAME#)
* Updates the data record to the specified dataset and database
* using *; for the record list.  On failure generates an error
* message, unlocks the database (assumes normal use of mode 1 for
* the DBOPEN) and aborts the program.
   CALL "DBUPDATE" USING !1
      !2
      DB-MODE-1
```

```
      DB-STATUS
      DB-SAME-LIST
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN.#
$DEFINE %DBUPDATE=
* %DBUPDATE(DB-BASENAME#,DS-M-SETNAME#,DA-M-SETNAME#
*      ,DL-ITEMLIST#)
* Updates the specified record to the specified dataset and
* database using an abbreviated item list and record area. On
* failure outputs an error message, unlocks the database (assumes
* normal use of mode 1 for DBOPEN) and aborts the program.
    CALL "DBUPDATE" USING !1
      !2
      DB-MODE-1
      DB-STATUS
      !4
      !3
    IF NOT DB-OKAY
     CALL "DBEXPLAIN" USING DB-STATUS
     CALL "DBUNLOCK" USING !1
        DB-DUMMY
        DB-MODE-1
        DB-STATUS
     STOP RUN.#
```

The V versions at the UAB School of Optometry are created by prefixing a V before the macro name and placing the following code prior to the DBEXPLAIN call in all macro definitions:

```
MOVE 9 TO V-CONTROL
CALL "VIEWIO" USING V-COM-AREA, @DATA-BUFFER
```

The VIEWIO procedure is a general interface to VPLUS which considerably simplifies normal access to VPLUS. Please see the screen usage standards for more information.

Please see Appendix A for a description of the naming conventions and required COPYLIB module for macro definitions' references.

DISCUSSION

Use of COBOL II macros has resulted in a consistent use of error detection and recovery while requiring less input by programmers. It also enables replacement of error handling code with more comprehensive reporting by modification of the macro definitions and recompilation with no examination required of individual source programs. See Appendix B for an example describing enhancement of the previously defined error reporting. Other advantages of the use of macros include reduction in source code disc space requirements, reduction in segment traps caused by the prior method of using PERFORMed routines with set-up MOVEs for similar access without a significant code space increase, simplification of calling sequences for IMAGE references with related time reduction for coding efforts, and simplification of debugging efforts as code is in-line which reduces the normal requirement of searching throughout an often large program source listing for various PERFORMed routines. REFERENCES

Thanks are due to the originators of COBOL II for including the macro processor capability (see Appendix A of Hewlett-Packard's COBOL II reference manual for a description of the macro processor, part number 32233-90001 and product number 32233A) and to Gerald Weinberg, author of High Level COBOL Programming, for his influence in making use of macro and preprocessors for solving programming problems. Also thanks to David Greer for his article IMAGE/COBOL: Practical Guidelines and Peter Somers for his article

Using COBOL, VIEW and IMAGE: A Practical Structured Interface for the Programmer in the 1982 San Antonio HP 3000 IUG Proceedings. Last but not least thanks to Tony Abruzzio of Union Camp Corporation for his contribution to the Montreal Contributed Library tape, COB-CAN, a COBOL callable cancel routine which provided most of the ideas utilized within Appendix B.

APPENDIX A

Following are standard declarations used in all programs which access a database. They are used to guarantee consistent data names and availability of all necessary fields.

```
01  DB-PARAMETERS.
    05  DB-ALL-LIST          PIC X(2) VALUE "@ ".
    05  DB-SAME-LIST         PIC X(2) VALUE "* ".
    05  DB-NULL-LIST         PIC X(2) VALUE "  ".
    05  DB-DUMMY             PIC S9(4) COMP VALUE ZERO.
    05  DB-PASSWORD          PIC X(8) VALUE "APASS;  ".
    05  DB-MODE-1            PIC S9(4) COMP VALUE 1.
    05  DB-MODE-2            PIC S9(4) COMP VALUE 2.
    05  DB-MODE-3            PIC S9(4) COMP VALUE 3.
    05  DB-MODE-4            PIC S9(4) COMP VALUE 4.
    05  DB-MODE-5            PIC S9(4) COMP VALUE 5.
    05  DB-MODE-6            PIC S9(4) COMP VALUE 6.
    05  DB-MODE-7            PIC S9(4) COMP VALUE 7.
    05  DB-MODE-8            PIC S9(4) COMP VALUE 8.
    05  DB-MODE-OTHER        PIC S9(4) COMP VALUE ZERO.
    05  DB-TEXT-LENGTH       PIC S9(4) COMP VALUE 72.
    05  DB-TEXT              PIC X(72).
    05  DB-STATUS.
        10  DB-CONDITION     PIC S9(4) COMP.
            88  DB-OKAY      VALUE 0.
            88  DB-LOGGING-DISABLED
                             VALUE 71.
            88  DB-BOF       VALUE 10.
            88  DB-EOF       VALUE 11.
            88  DB-BOC       VALUE 14.
            88  DB-EOC       VALUE 15.
            88  DB-MISSING   VALUE 17.
            88  DB-CHAIN-HEAD
                             VALUE 44.
            88  DB-GET-OKAY  VALUE 0, 10, 11, 14, 15, 17.
        10  DB-STAT2         PIC S9(4) COMP.
        10  DB-STAT3-4       PIC S9(9) COMP.
        10  DB-CHAIN-LENGTH  PIC S9(9) COMP.
            88  DB-EMPTY-CHAIN
                             VALUE ZERO.
        10  DB-STAT7-8       PIC S9(9) COMP.
        10  DB-STAT9-10      PIC S9(9) COMP.
```

In addition to the above defined standard declarations all programs define the following data values for database access on an as needed basis:

(Continued on next page.)

```
01  DB-BASENAME              PIC X(10) VALUE "  BASENAME".
01  DS-D-BASEDETAIL          PIC X(16) VALUE "D-BASEDETAIL    ".
01  DS-M-BASEMASTER          PIC X(16) VALUE "M-BASEMASTER    ".
01  DS-A-BASEAUTOMATIC       PIC X(16) VALUE "A-BASEAUTOMATIC ".
01  DR-D-BASEDETAIL.
COPY BASEDETA.
    (Data declarations for @; reference to dataset, normally
```

```
    retrieved from a COBOL COPYLIB file named COPYLIB.COBOL.)
01  DR-M-BASEMASTER.
COPY BASEMAST.
01  DA-D-BASEDETAIL.
COPY BASEDE1A.
    (Data declarations for an item list reference to dataset,
    normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DL-D-BASEDETAIL.
COPY BASEDE1L.
    (Item list declarations for an item list reference to
    dataset, normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DE-LOCKSPECIFICATION.
COPY LOCKSP1.
    (Lock specifications for locking on an item value, etc.,
    normally retrieved from a COBOL COPYLIB file named
    COPYLIB.COBOL.)
01  DI-KEYVALUE              PIC X(16) VALUE "KEYVALUE        ".
    (This is a key name for DBFIND or DBGET access.)
01  DV-KEYVALUE              PIC 9(6) DISPLAY.
    (This is the search value for use with searches on data
    item referenced by DI-KEYVALUE.)
```

APPENDIX B

Additional macros have been defined to enhance the error reporting of the initial version of the IMAGE database access macros. They add the capability of logging all detected errors to the system console as well as to the user's terminal. This has proven especially useful for debugging of problems for remote users. The database parameters COPYLIB was modified to include the following at the end of its definition:

```
05  ERROR-REPORT-A.
    10  ER-DATE               PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-TIME               PIC X(6) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-DEVICE             PIC ZZZ9 VALUE 0.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-USER               PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE ".".
    10  ER-ACCOUNT            PIC X(8) VALUE SPACES.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-STATUS             PIC ZZZ9- VALUE 0.
    10  FILLER                PIC X VALUE SPACE.
    10  ER-SUBSYSTEM          PIC X(8) VALUE "IMAGE   ".
05  ERROR-REPORT-B.
    10  ER-PROGRAM            PIC X(35) VALUE SPACES.
    10  FILLER                PIC XXX VALUE " @ ".
    10  ER-PARAGRAPH          PIC X(30) VALUE SPACES.
```

The following macro is invoked at the initiation of each program to define the logon device, user name, and program:

```
$DEFINE %DBINIT=
* %DBINIT(LISTAPPT.PROG.SCHOOL#)
* Fills in the current date, time of day, logon device, user
* name, and program name for use in generation of error messages.
    CALL INTRINSIC "WHO" USING //,
        //,
        //,
        ER-USER,
        //,
        ER-ACCOUNT,
        //,
        DB-CONDITION
    MOVE DB-CONDITION TO ER-DEVICE
```

```
MOVE CURRENT-DATE TO ER-DATE
MOVE TIME-OF-DAY TO ER-TIME
MOVE "!!" TO ER-PROGRAM.#
```

Each of the macro definitions for database access is modified to place the following code prior to each call to DBEXPLAIN:

```
MOVE DB-CONDITION TO ER-STATUS
DISPLAY ERROR-REPORT-A UPON CONSOLE
DISPLAY ERROR-REPORT-B UPON CONSOLE
```

Optionally the programmer may include code at the beginning of each paragraph which will place the name of the paragraph in the variable ER-PARAGRAPH. The recommended method for this is to use the following macro definition:

```
$DEFINE %AT=
* %AT(CONTINUE-PROCESSING#)
* Places the current paragraph name into a variable for reporting
* of error messages and easing the debugging process.
    MOVE "!!" TO ER-PARAGRAPH#
```

Use of this macro as standard practice also provides an excellent method of providing program trace and profiling capabilities by replacement of the macro definition. A simple example to show the execution path of a program would use a macro definition as follows:

```
$DEFINE %AT=
* %AT(CONTINUE-PROCESSING#)
* Places the current paragraph name into a variable for reporting
* of error messages and easing the debugging process and also
* lists the current active paragraph.  (Note:  full value of this
* would require resetting of the paragraph name following return
* from each PERFORMed paragraph.)
    MOVE "!!" TO ER-PARAGRAPH
    DISPLAY ER-PARAGRAPH UPON CONSOLE#
```

When it is desirable to reduce the code requirements a null macro which generates no code could replace the normal definition. Any number of functions become practical when this scenario becomes accepted practice.

Jerrel Baxter graduated cum laude from Birmingham-Southern College, a United Methodist Church supported liberal arts college located in Birmingham, Alabama, in 1975 with a Bachelor of Science degree in Mathematics. Following graduation he was employed by BSC as a computer programmer and instructor in computer science.

In 1978 he became manager of data processing for Argo and Company, a General Electric industrial electrical parts distributor located in Birmingham, Alabama, and Southern Carbon Brush Company, a manufacturer of carbon brushes for electric motors, generators, etc.

Following stabilization of the operation at Argo and Company, he left in 1981 to become manager of data processing for American Intermedical Resources, providers of respiratory therapy department operations for hospitals throughout the United States, computerized electrocardiogram analysis, computerized pulmonary function and arterial blood gas analysis, and equipment rental for home-based health care.

In December of 1982, he became lead systems analyst for the School of Optometry of the University of Alabama in Birmingham, a new HP 3000 Series 44 installation. Currently he is heavily involved in computerization of the clinic operations of the school which see approximately 30,000 patients each year for vision tests by optometry students and necessary spectacle and/or contact lens fitting.

He has provided consulting and programming services to the North Alabama Conference of the United Methodist Church and to Innovest Systems, Inc. (provides investment advisory services to institutional investors and recently to brokers of E. F. Hutton).

As an adjunct instructor at Birmingham-Southern College he has taught courses in COBOL programming, FORTRAN programming, BASIC programming, APL programming, and Data Structures.

The Ins and Outs of Pascal/3000 I/O

Susan R. Kimura
Hewlett-Packard

Numerous questions usually arise when a user first encounters I/O in Pascal. Some of these questions are:

- Do I have to use GET and PUT to do my I/O?
- What is the deferred GET all about?
- How exactly does EOLN work?
- How do READ and WRITE work?
- Why do READ and WRITE take longer than MPE's FREAD and FWRITE?

This paper addresses these questions. The discussion focuses on the standard Pascal file type called textfile. The first part discusses textfiles in general. The second part discusses the use of READs and WRITEs of textfiles. Some comparison with MPE's FREADs and FWRITEs is also made.

TEXTFILES

Pascal defines a file type as a structure which consists of a sequence of components of the same type. The following discussion focuses on a Pascal standard file type called TEXT. A variable of type TEXT is called a textfile. A textfile is a logical file which consists of components of type CHAR and is structured into lines separated by line markers. It is accessed sequentially. A textfile also has associated with it a buffer variable and a current component. The buffer variable is denoted f^ for a textfile f. It may be previewed prior to a read operation or modified prior to a write operation.

There are two primitives, GET(f) and PUT(f), which operate on a textfile. These primitives are used in conjunction with the buffer variable f^ to perform input and output. GET(f) is used for input and PUT(f) for output. The GET operation, as defined by Jensen and Wirth [3], advances the current file position to the next component and moves that component into the buffer variable. The PUT operation writes the contents of the buffer variable to the current component and advances to the next component.

The procedures RESET(f) and REWRITE(f) are provided to open a file for sequential I/O. RESET(f) opens the file f for read-only access. The file is positioned at the first component and a GET is performed. REWRITE(f) opens the file f for write-only access. The file is positioned at the first component. The file buffer variable f^ is undefined.

Note that this definition of RESET may cause problems for input from a terminal. If a GET, as defined above, is performed on the RESET a physical read from the terminal must be done in order to fill the buffer variable. The program is then paused for input from the terminal before the user has requested an input operation. The deferred GET, as defined in the HP Standard [2] and used by Pascal/3000, is a method which provides a solution to this problem. With the deferred GET the buffer variable is not filled on a GET. However, on the following reference to the buffer variable the current component is moved to the buffer variable and the file position is advanced to the next component. In other words, after a RESET the buffer variable f^ is undefined until an operation which references the buffer variable is requested. At that time the first component is moved to the buffer variable and the file position is advanced. An operation which would fill the buffer variable would be a reference to the buffer variable f^, or a call to READ(f,v), EOF(f), or EOLN(f).

EOF(f) and EOLN(f) are functions which indicate when the end-of-file has been

reached and when the end-of-line has been reached, respectively. EOF(f) is true when there are no more components to be read from file f. EOLN(f) is true when the current position is beyond the last component of a line. At this point the buffer variable f^ contains a blank.

An overview of the Pascal/3000 internals associated with a file would be useful at this point. When a file is declared, a block of storage is allocated which is used to keep track of the state of the file. This block of storage is called the file control block and contains information such as a readable flag, a writeable flag, an end-of-line flag, an end-of-file flag, a get flag, the current record length, the MPE file number, the file buffer variable, and the current position. Another block of storage is allocated which buffers the input or output of a physical I/O operation. This block of storage is called the file buffer. When an I/O operation is requested, the Pascal/3000 run-time library accesses and updates the information in the file control block. In the case of an input operation a record may be read from the file into the file buffer and characters moved from the file buffer to the buffer variable. In the case of an output operation characters may be moved from the buffer variable to the file buffer and the contents of the file buffer written to the file.

Consider the following program which uses GET(f) for reading characters from a textfile f. The physical file associated with f contains only one line with the sequence 'AB'.

```
        PROGRAM pascal (f);
        VAR
           f : TEXT;
           c : CHAR;
           b : BOOLEAN;
        BEGIN
 (1)    RESET(f);
 (2)  c := f^;
 (3)  GET(f);
 (4)  c := f^;
 (5)  GET(f);
 (6)  b := EOLN(f);
 (7)  GET(f);
 (8)  b := EOF(f);
      END.
```

The results after the execution of each statement are:

	c	b	feof	feoln	fcpos	fchar	fgetok
(1)	?	?	false	true	0	?	true
(2)	'A'	?	false	false	1	'A'	false
(3)	'A'	?	false	false	1	'A'	true
(4)	'B'	?	false	false	2	'B'	false
(5)	'B'	?	false	false	2	'B'	true
(6)	'B'	true	false	true	2	' '	false
(7)	'B'	true	false	true	2	' '	true
(8)	'B'	true	true	true	2	' '	false

Feof, feoln, fcpos, fchar and fgetok are variables in the file control block. Feof is the end-of-file flag, feoln the end-of-line flag, fcpos the current character position index, fchar the buffer variable and fgetok the get flag which indicates that the buffer variable may be updated.

On the RESET(f) in (1), the textfile f is opened for read-access. Fcpos is set to the first position, which is 0. Fgetok is set to true but fchar is still undefined. When the buffer variable is referenced in (2) an FREAD is done, filling the file buffer with the line 'AB'. Fchar is filled with the first character, 'A', and fcpos is updated. When GET(f) is called in (3), fgetok is set to true, but fchar is not updated. Statements (4) and (5) are similar to statements (2) and (3). When EOLN(f) called in in (6), the fcpos is beyond the last character. Consequently, fchar is filled with a blank as defined in Pascal and feoln is set to true. The GET(f) in (7) is used to get past the end-of-line marker by setting fgetok to true. Finally, the call to EOF(f) in (8) triggers another FREAD which returns an end-of-file condition and feof is set to true.

Writing to a file using PUT(f) is a straightforward case. Consider the following program which writes to the textfile f:

```
        PROGRAM pascal (f);
        VAR
           f : TEXT;
        BEGIN
 (1)    REWRITE(f);
 (2)    f^ := 'a';
 (3)    PUT(f);
 (4)    f^ := 'b';
 (5)    PUT(f);
        END.
```

The results after the execution of each statement are:

	fcpos	fchar
(1)	0	?
(2)	0	'a'

```
(3)      1       'a'
(4)      1       'b'
(5)      2       'b'
```

On the REWRITE(f) in (1) the textfile f is opened for write-access. Fcpos is set to the first position. The buffer variable fchar is undefined until the literal 'a' is assigned to it in (2). On the PUT(f) in (3) the contents of fchar is moved to the file buffer and fcpos is updated. Statements (4) and (5) are similar to statements (2) and (3). When the end of the program is encountered, the file buffer is automatically flushed. An FWRITE is done to the file f which now contains the sequence 'ab'.

READ, READLN, WRITE, WRITELN

While GETs and PUTs may be used for textfile I/O, they are not the most efficient method. To facilitate textfile I/O the predefined procedures READ, READLN, WRITE, and WRITELN are provided.

The forms of the READ statement are:

```
READ(f,v);
READ(f,v1,...,vn);
READLN(f);
READLN(f,v);
READLN(f,v1,...,vn);
```

The forms of the WRITE statement are:

```
WRITE(f,e);
WRITE(f,e1,...,en);
WRITELN(f);
WRITELN(f,e);
WRITELN(f,e1,...,en);
```

These procedures allow the user to read variables and write expressions of type char, integer, real, longreal, boolean, user-defined enumeration, PAC, and string. 1 The f parameter may be omitted. In this case the standard Pascal textfiles INPUT and OUTPUT are used for READs and WRITEs, respectively. The parameter v1,...,vn indicates that an arbitrary number of variables may be read. Likewise, the parameter e1,...,en indicates that an arbitrary number of expressions may be written. They need not all be of the same type. Furthermore, when reading or writing integer, real, longreal or user-defined enumeration, internal conversions occur. These conversions, from ascii to binary representation in the case of a read, or binary to ascii representation in the case of a write, greatly simplify the work for the user.

On a READ or READLN a sequence of characters which conform to the syntax of the type of the variable are read and converted from ascii to binary, if necessary. After a READ the file position is set after the last character read on the current line. After a READLN, any remaining characters and the end-of-line marker in the current line are skipped over and the file position is set at the start of the following line.

Conversely, on a WRITE or WRITELN, an expression is converted from binary to ascii, if necessary, and written to the file buffer. After a WRITE the file position is set to the position after the last character written. On a WRITELN an end-of-line marker is written after the last character 2 and the file buffer is written to the file using an FWRITE.

The write statements also allow formatting of output by specifying a field width parameter m. For reals and longreals, a decimal place parameter n may also be specified. The forms of the write expression are:

```
e
e:m
e:m:n
```

If no formatting is specified, a default field width is used. If m is specified, the expression is written in the field width specified with right-justification. For reals and longreals, if n is specified, the expression is written in fixed-point format with n decimal digits.

Consider the simplest case of reading in a sequence of characters into a variable c of type CHAR with the following statements from a textfile f which contains two lines, 'AB' and 'DEF'.

```
          PROGRAM pascal (f);
          VAR
             f : TEXT;
             c : CHAR;
             b : BOOLEAN;
          BEGIN
(1)       RESET(f);
(2)       READ(f,c);
(3)       READ(f,c);
(4)       b := EOLN(f);
(5)       READ(f,c);
(6)       READ(f,c);
          END.
```

Because the characters are being read one at a time, this sequence of statements is equivalent to:

```
          PROGRAM pascal (f);
          VAR
             f : TEXT;
             c : CHAR;
             b : BOOLEAN;
          BEGIN
(1)       RESET(f);
(2)       c := f^; GET(f);
(3)       c := f^; GET(f);
(4)       b := EOLN(f);
```

```
(5)     c := f^; GET(f);
(6)     c := f^; GET(f);
        END.
```

The results after execution of each statement are:

	c	b	feoln	fcpos	fchar	fgetok
(1)	?	?	true	0	?	true
(2)	'A'	?	false	1	'A'	true
(3)	'B'	?	false	2	'B'	true
(4)	'B'	true	true	2	' '	false
(5)	' '	true	true	2	' '	true
(6)	'D'	true	false	1	'D'	true

The textfile f is opened for read-access on the RESET(f) in (1). Fcpos is set to the first position. Fgetok is set to true but fchar is still undefined. When READ(f,c) is called in (2), an FREAD is done, filling the file buffer with 'AB'. Fchar is filled with the first character, 'A', and fcpos is updated. The contents of fchar is then assigned to the variable c. In addition, because a GET(f) is part of the READ, fgetok is set to true. Statement (3) is similar to statement (2). When EOLN(f) is called in (4) fchar is filled with a blank because fcpos is beyond the last character. The next call to READ(f,c) in (5) the blank in fchar is assigned to the variable c and fgetok is set to true. Finally, on the READ(f,c) in (6), another FREAD is done, filling the file buffer with 'DEF'. Fcpos is initialized to the first position, fchar is filled with 'D' and its contents assigned to the variable c.

If the READ(f,c) in (5) is replaced by READLN(f) with no variable to read, this would be equivalent to a GET(f) and the corresponding result would be:

	c	b	feoln	fcpos	fchar	fgetok
(5)	'B'	true	true	0	' '	true

In this case, fcpos is set to the first position, fgetok is set to true and the variable c remains unchanged.

As stated above READ(f,v) allows the user to read not only characters but also integers, reals, enumerated types, PACs and strings. Internally, when reading a variable of type INTEGER from a textfile f, the following sequence of events occur in the Pascal run-time routine for reading an integer:

(1) A procedure is called to verify that the file is open. This procedure searches a linked list of opened files. If the file is not found on the list, an error is reported.

(2) The read-access flag in the file control block is checked to verify that the file has read-access. If it does not, an error is reported.

(3) A procedure is called to verify that an end-of-file condition does not exist. A physical read using FREAD may occur. If an end-of-file is encountered, an error is reported.

(4) The buffer variable is scanned to skip over initial blanks. Characters may be moved from the file buffer to the buffer variable. FREADs may occur if end-of-line markers are encountered.

(5) A procedure is called to verify that an end-of-file condition was not encountered while skipping blanks. If so, an error is reported.

(6) A string of digits is moved into an internal buffer until a non-digit or end-of-line marker is encountered. The digits are moved from the file buffer, to the buffer variable, and then to the internal buffer.

(7) The compiler library routine EXTIN' is invoked to do the conversion to binary.

An analogous sequence of events occur when reading a real or longreal variable with the exception that reading the string of digits is more complex due to the decimal point and scale factor which may be present.

This run-time routine shows how much overhead there is compared with doing an FREAD. The user who uses FREAD has direct control over the state of his file. Consequently, the checks for whether the file is open and has read-access are not necessary. Since the Pascal

run-time routine does not know the dynamic flow of a user program these checks are necessary. On the other hand, the user who uses FREAD is responsible for doing the ascii to binary conversion. This is automatically done by the run-time routine. The primary overhead in the Pascal run-time library, is the movement of characters from the file buffer, to the buffer variable, then to the internal buffer. These steps are done in order to maintain the file control block in the correct state at all times.

Now consider the case of writing an integer to a textfile f. The following events occur:

(1) A procedure is called to verify that the file is open. If it is not, an error is reported.

(2) The write-access flag in the file control block is checked to verify that the file has write-access. If it does not, an error is reported.

(3) A check is made that a valid field width has been requested. If the field width is not valid, an error is reported.

(4) The compiler library routine IN-EXT' is invoked to convert the binary representation to an ascii string.

(5) The correct field width to use is calculated. The user-specified field width may be over-ridden if the converted string does not fit within the user-specified width.

(6) A WRITELN is done if the converted string does not fit on the current line.

(7) A procedure is called to write out the converted string. If the converted string is shorter then the determined field width, an appropriate number of blanks are inserted so that the string will be right-justified within the determined field width. The characters are moved from the string to the file buffer and the current position index is updated.

As demonstrated above, there is also some overhead in writing an integer to a textfile. Checks must be done to verify that the file is in the proper state. The conversion of the integer from binary to ascii representation must be done. Finally, the ascii representation must be moved to the file buffer. For the user using FWRITE, the check that the file is opened for write-access is not necessary. However, the conversion from binary to ascii, as well as formatting of output, which are taken care of by the run-time routine, must be done by the user.

Finally, consider the case of reading a variable of type PAC from a textfile. The following events occur:

(1) A procedure is called to verify that the file is open. If it is not open, an error is reported.

(2) The read-access flag in the file control block is checked to verify that the file has read-access. If it does not, an error is reported.

(3) A procedure is called to check for an end-of-line condition. An FREAD may occur. If an end-of-file condition is encountered, an error is reported. If an end-of-line condition is encountered, the PAC variable is blank filled and control is returned to the user program.

(4) If an end-of-line condition was not encountered, characters are moved from the file buffer to the buffer variable, then to the PAC variable, one character at a time, until the variable has been filled or an end-of-line condition is encountered.

(5) If the PAC variable was not filled, the remaining character positions are blank-filled.

As with the other run-time routines, checks must be made to verify that the file is opened with the appropriate access. When checking for the end-of-line condition, a physical read using FREAD into the file buffer may occur. Procedures are called to move the characters from the file buffer into the buffer variable and then into the PAC variable. Finally, the variable is blank-filled if necessary. This description shows that reading a PAC variable, which may be thought of as similar to an FREAD of a PAC variable, is more complex. The primary overhead is in accessing the routines which maintain the state of the file control block which are not neccessary for user FREADs.

To summarize, a great deal of overhead is involved in accesssing textfiles because the state of the file control block must be maintained, especially with respect to the buffer variable and the current position. The fact that textfiles have end-of-line markers complicate reading and writing of textfiles. On the other hand, the use of textfile I/O eliminates the need for the user to use or write his own conversion routines when reading and writing

integer, real, longreal, boolean, and user-defined enumeration. Furthermore, formatting of output is done for the user using the write routines.

The author hopes that this discussion has eliminated some of the "mystery" associated with textfile I/O and that this information may aid in the user's deveopment of Pascal programs.

Footnotes

1 In Pascal/3000 PAC is used to denote the type PACKED ARRAY [1..N] OF CHAR. String is a predefined packed structured type consisting of components of type CHAR. It has a maximum length but its actual length may vary dynamically at run time.

2 On the HP3000 the end-of-line marker is a logical marker. There is no actual end-of-line marker written to the file buffer.

Bibliography

[1] American National Standard Pascal Computer Programming Language. (1983). ANSI/IEEE 770X3.97-1983.

[2] Hewlett-Packard Standard Pascal Report. (1983). Release 2.

[3] Jensen, Kathleen and Wirth, Niklaus. (1975). Pascal User Manual and Report. Springer-Verlag. New York.

[4] Pascal/3000 Reference Manual. (1981). Hewlett-Packard.

Acknowledgement

I would especially like to thank my colleague Chris Maitz for his expertise and advice. I would also like to thank my project manager, Jean Danver, and my colleages, Jon Henderson and Pat Miyamoto, for proof-reading this paper.

Biographical Sketch

Name : Susan R. Kimura

Title : Member of Technical Staff

Employer : Hewlett-Packard Computer Languages Lab

Job Responsibilities : Involved in current product engineering on the COBOLII and Pascal/3000 compilers since 1980.

Education: BA, University of Hawaii, Psychology MA, University of Hawaii, Japanese Linguistics MS, University of Wisconsin, Computer Sciences

Marital Status : Married with one child

Systematic Redesign
Modifying Object Code

Phil Curry
Alvin Community College

Many different types of files may be maintained on a computer system, right? Well in actuality this statement is false. There are only two types of files, data files and program files. The data structures and contents of the files are basically immaterial to the type of file it is. The way a file is being used determines the type of file it is. Let us prove this concept by looking at several files that would be typically found on a computer system.

A :LISTF command is done to examine file information on a particular file.

```
:LISTF DATA,2
ACCOUNT= ISIS        GROUP= MISC

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

DATA              80B  FA           4          4   3        3  1  1


The file contents are as follows:

R.P. Gee              1401 Printer Street         White Plains    NY
Anne C. Cobol         P.O. Box X3J4               Boston          MA
N.C. Seay             4 Tran Street               Los Angeles     CA
Flop E. Disc          256 Sector                  Half Track      IL
```

What does the file contain? The answer is fairly easy, the file contains data.

A :LISTF command is done to examine file information for another file.

```
:LISTF FSOURCE,2
ACCOUNT= ISIS        GROUP= MISC

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

FSOURCE           80B  FA          14         14   3        6  1  1


The file contents are as follows:

$CONTROL USLINIT,SOURCE,WARN
$TITLE "SUMMATION PROGRAM"
      PROGRAM SUMM
```

```
      IMPLICIT INTEGER (A-Z)
      SUM=0
C
   10 DISPLAY "ENTER NUMBER: "
      ACCEPT NUMBER
      IF (NUMBER.EQ.-1) GO TO 20
      SUM=SUM+NUMBER
      GO TO 10
   20 DISPLAY "The sum of the numbers is: ", SUM
      STOP
      END
```

What does the file contain? Again, the file contains data. The file does not contain a program as some would answer. This file is used as input data to a Fortran compiler which produces a USL file as output. When a :PREP command is used at the MPE level or the -PREPARE command is used in the Segmenter, the USL file is used as data to the Segmenter and a file containing object code is produced.

Looking at one other file on the system, a :LISTF command shows the following:

```
:LISTF FOBJECT,2
ACCOUNT= ISIS        GROUP= MISC

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

FOBJECT   PROG    128W  FB           5          5   1        6  1  1
```

The file contents (as viewed with FCOPY) are as follows:

```
:RUN FCOPY.PUB.SYS

HP32212A.3.17 FILE COPIER (C) HEWLETT-PACKARD CO. 1982

>FROM=FOBJECT;TO=;OCTAL;CHAR

FOBJECT RECORD 0 (%0, #0)

00000: 004600 000001 000003 000001 ........
00004: 000002 001440 000000 177777 ... ....
00010: 000004 000000 000000 177777 ........
00014: 000000 000003 000001 177777 ........
00020: 000000 000000 000000 000000 ........
00024: SAME: TO 000034-1
00034: 140400 000124 000000 000000 ...T....
00040: SAME: TO 000050-1
00050: 000400 000600 001000 000171 .......y
00054: 000371 000000 000000 177310 ........
00060: 177000 177000 177000 000002 ........
00064: 001362 000576 001374 000000 ...~....
00070: 002003 117103 002601 001400 ...C....
00074: 152703 043117 041112 141075 ..FOBJ.=
00100: 052040 002001 000767 000400 T ......
00104: 000200 001120 000601 046511 ...P..MI
00110: 051503 020040 020040 044523 SC    IS
00114: 044523 020040 020040 000000 IS    ..
00120: 000000 000000 177000 177000 ........
00124: 177004 177006 000007 000006 ........
00130: 177511 000017 177310 000001 .I......
00134: 000042 000001 000000 000000 ."......
00140: 000642 000362 177310 177310 ........
00144: 177310 000000 000000 000000 ........
00150: 177310 000001 000112 000000 .....J..
```

```
00154: 000000 000003 001400 152703 ........
00160: 177730 000000 000001 001544 .......d
00164: 001560 001600 000001 000000 .p......
00170: 000000 000713 000400 130564 .......t
00174: 001400 064445 000000 000000 ..i%....

FOBJECT RECORD 1 (%1, #1)

00000: 003000 002400 177777 000000 ........
00004: SAME: TO 000200-1

< CONTROL Y >

2 RECORDS PROCESSED *** 0 ERRORS
```

What kind of file is this? The answer to this question is not as easy as the others. The file could be a data file or a program. We must run :LISTDIR2.PUB.SYS to answer this question.

```
:RUN LISTDIR2.PUB.SYS

LISTDIR2 C.01.00 (C) HEWLETT-PACKARD CO., 1977
TYPE 'HELP' FOR AID

>LISTF FOBJECT;PASS
**********
FILE: FOBJECT.MISC.ISIS

FCODE: PROG             FOPTIONS: STD,BINARY,FIXED
BLK FACTOR: 1           CREATOR: MANAGER
REC SIZE: 256(B)        LOCKWORD:
BLK SIZE: 128(W)        SECURITY--READ:    ANY
EXT SIZE: 6(S)                  WRITE:   ANY
# REC: 5                       APPEND:  ANY
# SEC: 6                       LOCK:    ANY
# EXT: 1                       EXECUTE: ANY
MAX REC: 5                    **SECURITY IS ON
MAX EXT: 1              COLD LOAD ID: %15106
# LABELS: 0             CREATED: THU, 23 JUN 1983
MAX LABELS: 0           MODIFIED: THU, 23 JUN 1983
DISC DEV #: 3           ACCESSED: THU, 23 JUN 1983
DISC TYPE: 0            LABEL ADR: %146665
DISC SUBTYPE: 9         SEC OFFSET: %1
CLASS: DISC             FLAGS: NO ACCESSORS
FCB VECTOR: %0

# SEG: 1                TOTAL DB: %3
STACK: %1440            DL: %0
MAXDATA: DEFAULT        CAP: IA,BA
>EXIT
```

Since the display does not indicate the file is loaded, this is a data file. The loader uses this file as data when a program is to be run that is not currently loaded on the system. Once the program is loaded, we would be correct in saying that this file contains a program. When the program is loaded, the file cannot be modified since the operating system has control over it. However, if the file is not being used for program execution, it may be accessed just as any other data file on the system.

The perpetual data concept should be quite obvious. The output of a process is data to another process. A person uses a terminal and a text editor to produce a file containing source code. This source code is data to a compiler which produces a USL file, and so on. Any of the files throughout the process may be accessed and modified before con- tinuing to the next logical step. What will be demonstrated is the taking of the output from the Segmenter, prepared object code, and modifying it before it is used as data to the loader.

DEBUG and DECOMP are used quite extensively in this paper. The reader is assumed to be familiar with DEBUG and the architecture of the HP 3000 including the structure of the data stack. You will learn quickly there are several tools which are invaluable. Several manuals which will be necessary are the Machine Instruction Set, System Reference, System Intrinsics and the SPL Reference manual along with the Instruction Decoding Pocket Guide. A copy of the System Tables Manual could also prove helpful.

Many people wonder why in the world would anyone want to modify object code. The only practical answer is "Any time a program needs to be modified and the source code is not available". If the source code is available, it would be much simpler to make the necessary changes to the code and create a new object file. All of us, however, have programs on our systems which we have no source code for. An example would be some of the programs found in the Contributed Software Library. Many times people will contribute only object code to make sure all changes to the source code are controlled at their site.

Many times though, we need to make changes to a program for which we have no source code. One example of this would be the run-time information placed in the the object code by the Segmenter. Looking at the listing produced by LISTDIR2 we can see there is one code segment, the total global-DB area is three words, the stack parameter used at run time is 1440 (octal), the DL size in words is zero, the maxdata parameter used at run time is the default (-1), and the capabilities of the program are IA and BA. The values mentioned are stored in record zero of the object code. The data structure for prepared object code can be found in the System Tables Manual in Chapter 10. There is also an excellent article in the proceedings for the 1978 HP User's Group North America meeting held in Denver which explains in a more narrative fashion the structure of object code and USL files.

Record zero of all object code contains the following:

Word (zero based)	Contents
0	Flags (Capabilities) Bit 7: BA 8: IA 9: PM 12: MR 14: DS 15: PH
1	Number of segments in the program
2	The number of words in the global-DB area of the program's run-time stack
5	The initial stack size
6	The initial DL size (zero if DL= not specified in PREP)
7	The maxdata specification (-1 if maxdata= not specified in PREP)

By knowing the above, any of the parameters which are set by the Segmenter when the object code is created may be changed by updating the appropriate word in the object code. One would not want to modify the number of segments unless you are going to add a segment to the object code (no easy task). The total global-DB value one would rarely want to change. By altering the total global-DB up, however, one can set aside some global area that the original program knows nothing about to be used by the patch being implemented or a patched in procedure.

Now that we know how to change the parameters set by the Segmenter, we need to know how to modify the instructions generated by the compiler. One way to do this is to use

the program DECOMP which is in the Contributed Software Library. DECOMP allows the dumping of object code, showing the op codes for the program instructions, and allows the modification of these instructions. Please notice that I stated modification of instructions. Other means must be taken for inserting instructions into object code. Deleting instructions is another matter. Any instruction can be quickly "deleted" by changing it to a NOP, which is the mnemonic for a "No operation" or do nothing instruction.

Let us look at a typical problem and a solution. We have a program which opens an Image data base with exclusive access to generate a report. There is a problem in that this program must be run when no one is accessing the Image data base. We do not have the source code for the program but we need to change the call to DBOPEN to use mode 5 rather than mode 3 to allow concurrent access to the data base.

First off, we will need to verify the fact that the DBOPEN is really using mode 3. The program calls DBEXPLAIN when a DBOPEN fails to show what happened. In the display we are shown that mode 3 was in fact used.

```
:RUN XTEST;LIB=P

OPEN ERROR OF DATA BASE

IMAGE ERROR AT %000010: CONDITION WORD = -1
DBOPEN, MODE 3, ON ISISDB
DATA BASE IN USE


END OF PROGRAM
:
```

If the program did not do this, we could use DBUTIL's SHOW command to verify that the program did indeed open the data base with a mode of 3.

Next we need to get a load map for the program to see where DBOPEN is called from in the program. The format for LMAPs can be found in the MPE Commands manual in appendix D.

```
:RUN XTEST;LMAP;MAXDATA=10000;LIB=P

PROGRAM FILE XTEST.MISC.ISIS

CANYDATE1          PROG 2  13    3 PSL  2  13    7
SEMESTER           PROG 2  11    3 PSL  0   2    4
ISISPREP           PROG 2   2    3 PSL  0   7    4
CVRI               PROG 2  14    2 PSL  2  20    7
CCOMPRESS          PROG 2  12    3 PSL  2  15    7
                           13    2
CFLAGS             PROG 2  10    2 PSL  2   7   14
                            7    1
SORTEND            PROG 0  23    3 SSL  0   1  206
C'PERFORM          PROG 0  22    3 SSL  0  11  232
SORTINITIAL        PROG 0  21    3 SSL  0  11  206
C'SORTERR          PROG 0  20    3 SSL  0  16  232
C'CLOSE            PROG 0  17    3 SSL  0   2  232
C'SORTINITIAL      PROG 0  16    3 SSL  0  11  231
C'ACCEPT           PROG 0  14    3 SSL  0  50  231
DBOPEN             PROG 2   3    3 SSL  2   2  276
DBOPEN             PSL  2  11    4 SSL  2   2  276
DBCLOSE            PROG 2  16    2 SSL  2  13  267
SORTINPUT          PROG 0  15    2 SSL  0   3  206
C'TST'             PROG 0  11    2 SSL  0  12  231
TERMINATE'         PROG 0  10    3 SSL  0   2   41
                            7    2
DBEXPLAIN          PROG 2   7    3 SSL  2   1  300
                            3    2
DBGET              PROG 2   2    2 SSL  2   2  272
C'ENDPAR           PROG 0  17    2 SSL  0   3  232
                           14    1
C'DISPLAY'FIN      PROG 0   6    3 SSL  0  36  231
                            6    2
                           13    1
C'DISPLAY'FC       PROG 0  12    1 SSL  0  37  231
C'DISPLAY'INIT     PROG 0   4    3 SSL  0  34  231
                            4    2
                           11    1
C'DISPLAY'L        PROG 0   5    3 SSL  0  33  231
                            5    2
                           10    1
C'EDITMOVEAN       PROG 0   6    1 SSL  0   2  231
C'WRITE'           PROG 0   5    1 SSL  0  20  232
SORTOUTPUT         PROG 0   4    1 SSL  0   2  206
C'EDITMOVEN        PROG 0   3    1 SSL  0   1  231
C'OPEN'            PROG 0  15    3 SSL  0  23  232
                            2    1
QUIT               PROG 0  15    1 SSL  0  21   16
                           10    0
QUIT               PSL  0  36   14 SSL  0  21   16
QUIT               PSL  0   5   10 SSL  0  21   16
QUIT               PSL  0  32    7 SSL  0  21   16
C'GOTO             PROG 0  12    2 SSL  0   4  232
                            7    0
COBOLTRAP          PROG 0   3    0 SSL  2  44  231

DEBUG              PROG 0   2    0 SSL  0  10   52
PRINT              PSL  0  40   14 SSL  0  35   45
PRINT              PSL  0   4   10 SSL  0  35   45
PRINT              PSL  0  12    4 SSL  0  35   45
WHO                PSL  0  37   14 SSL  0  30   45
DLSIZE             PSL  0  11   10 SSL  0  25  100
GENMESSAGE         PSL  0  10   10 SSL  0  11   65
PRINTFILEINFO      PSL  0   7   10 SSL  0   2   46
PRINTFILEINFO      PSL  0  23    4 SSL  0   2   46
FOPEN              PSL  0   6   10 SSL  0   1    4
FOPEN              PSL  0  16    4 SSL  0   1    4
```

```
RTOI'           PSL  0  40   7 SSL  0 102 203
COMMAND         PSL  0  37   7 SSL  0   1  25
FMTDATE         PSL  0  36   7 SSL  0  14  65
CLOCK           PSL  0  35   7 SSL  0  16  35
CALENDAR        PSL  0  34   7 SSL  0  17  35
FMTCALENDAR     PSL  0  33   7 SSL  0  15  65
CONVERTDATE     PSL  0  31   7 SSL  0  23  65
MYCOMMAND       PSL  0  30   7 SSL  0  27  45
FCONTROL        PSL  0  27   7 SSL  0   6 104
FCONTROL        PSL  0  20   4 SSL  0   6 104
FSETMODE        PSL  0  26   7 SSL  0   4 104
FREAD           PSL  0  25   7 SSL  0   4  15
FREAD           PSL  0  17   4 SSL  0   4  15
FWRITE          PSL  0  24   7 SSL  0   3  15
FWRITE          PSL  0  13   4 SSL  0   3  15
FREADDIR        PSL  0  25   4 SSL  0   1  15
TERMINATE       PSL  0  24   4 SSL  0   1  41
DFIX            PSL  0  21   4 SSL  0   3 205
DFLOAT'         PSL  0  15   4 SSL  0 151 203
EXTIN'          PSL  0  14   4 SSL  0  41 205

301 302 303 304

OPEN ERROR OF DATA BASE

IMAGE ERROR AT %000010: CONDITION WORD = -1
DBOPEN, MODE 3, ON ISISDB
DATA BASE IN USE


END OF PROGRAM
:
```

DBOPEN is called in the program code from segment 3 and is at external segment transfer table entry 3. By looking at the LMAP, we can tell that mode 5 can be used since there are no calls to DBLOCK or any of the Image intrinsics which modify the data base.

Next, DECOMP is run to find where in segment 3 the PCALs to DBOPEN occur. The first command entered is to determine where the segment transfer table begins in the code segment.

```
:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XTEST
TYPE 'HELP' FOR ASSISTANCE.

-F 3.
  3.500   100601  SEGMENT TRANSFER TABLE (PL-%23)  SORTEND
  3.501   104600  SEGMENT TRANSFER TABLE (PL-%22)  C'PERFORM
  3.502   104601  SEGMENT TRANSFER TABLE (PL-%21)  SORTINITIAL
  3.503   107200  SEGMENT TRANSFER TABLE (PL-%20)  C'SORTERR
  3.504   101200  SEGMENT TRANSFER TABLE (PL-%17)  C'CLOSE
  3.505   104602  SEGMENT TRANSFER TABLE (PL-%16)  C'SORTINITIAL
  3.506   111600  SEGMENT TRANSFER TABLE (PL-%15)  C'OPEN'
  3.507   124202  SEGMENT TRANSFER TABLE (PL-%14)  C'ACCEPT
  3.510   105576  SEGMENT TRANSFER TABLE (PL-%13)  CANYDATE1
  3.511   106576  SEGMENT TRANSFER TABLE (PL-%12)  CCOMPRESS
  3.512   101177  SEGMENT TRANSFER TABLE (PL-%11)  SEMESTER
  3.513   101042  SEGMENT TRANSFER TABLE (PL-%10)  TERMINATE'
  3.514   100534  SEGMENT TRANSFER TABLE (PL-%7)  DBEXPLAIN
  3.515   117202  SEGMENT TRANSFER TABLE (PL-%6)  C'DISPLAY'FIN
  3.516   115602  SEGMENT TRANSFER TABLE (PL-%5)  C'DISPLAY'L
  3.517   116202  SEGMENT TRANSFER TABLE (PL-%4)  C'DISPLAY'INIT
```

```
  3.520   101132  SEGMENT TRANSFER TABLE (PL-%3)  DBOPEN
  3.521   103577  SEGMENT TRANSFER TABLE (PL-%2)  ISISPREP
  3.522   000000  SEGMENT TRANSFER TABLE (PL-%1)
  3.523   040023  STT LENGTH = %23
```

The last instruction will be 0.477 since the segment transfer table starts at 3.500, so we look for a PCAL to DBOPEN in 0.0 through 0.477.

```
-F "PCAL  DBOPEN",0/477

  3.10    031003  2.    PCAL  DBOPEN

-EXIT

END OF PROGRAM
:
```

DBOPEN is called from one place in the program, segment 3 offset 10. Now we need to run the program and invoke Debug to determine a memory location to use which contains the value 5 to reference in the DBOPEN for the mode parameter.

```
:RUN XTEST;LIB=P;DEBUG


*DEBUG* 0.0
```

A breakpoint is set at the call to DBOPEN.

```
?B 3.10
?R


*BREAK* 3.10
```

Next we can look at the program segment to verify we broke at the correct point in the program.

```
?DP-5,6,C
P-5       031002  PCAL STT 2
P-4       040022  LOAD P+22
P-3       040022  LOAD P+22
P-2       040022  LOAD P+22
P-1       040022  LOAD P+22
P+0       031003  PCAL STT 3
```

Next we look at the run time stack to see what was placed on the stack in preparation for the call to DBOPEN. DBOPEN has 4 parameters so we look at the top 4 items on the stack. Notice also in the dump of the program segment above that we can see where the 4 loads to top of stack occur.

```
?D S-3,4
S-3       000513 000535 000540 000471
```

Since the intrinsic DBOPEN uses call by reference for all it's parameters, we know that these values on the stack are memory addresses and not actual values.

The parameters are located in the stack as follows:

Location	Parameter for DBOPEN
S-3	BASE
S-2	PASSWORD
S-1	MODE

```
    S-0                STATUS
```

We can verify this by displaying the contents of memory at these locations.

```
    ?D 513,10,A
    DB+513       ISISDB
    ?D 535,10,A
    DB+535       ; ..............
    ?D 540,I
    DB+540       +00003
```

We know that the DBOPEN is for the data base ISISDB using the password of the creator (;) with an open mode of exclusive access, mode 3.

Next we look at the memory locations around the mode parameter to see if there is a location containing a 5 which can be substituted. We look for an alternate memory reference rather than changing the value of DB+540 since this location could be referenced elsewhere in the program and changing the contents could produce unpredictable results.

```
    ?D 530,20,I
    DB+530       +08224 +08224 +08224 +08224 +08224 +15136 +00001 +00002
    DB+540       +00003 +00004 +00005 +00006 +00007 +00000 +00000 +00000
```

A pattern of values can be found by looking at DB+536 through DB+544. The values contained in these locations are 1 through 7. Since this pattern is found, we will assume that these are being used as constants in the program for the Image calls. This could be verified by setting breakpoints at calls to DBFIND, DBGET and DBCLOSE and examining the locations passed to these calls, however since we are highly certain these values are alright to use as constants, we will continue.

Next we will modify the contents of S-1, the parameter for the mode, from DB+540 which contains the value 3, to DB+542 which contains the the value 5.

```
    ?MS-1
    S-1        000540 :=542
    ?R

    Do you wish to extract only on-campus sections?
```

As we can see the data base opened and the program continued executing.

We have now found out what needs to be changed in the call to DBOPEN. Now we need to patch the object code to call DBOPEN using the address DB+542 for the mode parameter rather than DB+540.

```
    :RUN DECOMP.LIB.SYS

    HP3000 DECOMPILER 6.1

    FILE NAME? XTEST
    TYPE 'HELP' FOR ASSISTANCE.

    -3.0
    FILE  XTEST.MISC.ISIS
    SEGMENT 3          LENGTH 524


      3.0      170400  ..    LRA   P- 000      <<=0>><-- Procedure Entry
      3.1      051604  S.    STOR  Q- 004
      3.2      031400  3.    EXIT  %0

      3.3      031002  2.    PCAL  ISISPREP
      3.4      040022  @.    LOAD  P+ 022      <<=26>>
      3.5      040022  @.    LOAD  P+ 022      <<=27>>
      3.6      040022  @.    LOAD  P+ 022      <<=30>>
      3.7      040022  @.    LOAD  P+ 022      <<=31>>
```

```
3.10    031003  2.    PCAL  DBOPEN
3.11    040020  @.    LOAD  P+ 020      <<=31>>
3.12    043700  G.    LOAD  S- 000,I
3.13    000106  .F    DELB, ZERO
3.14    001706  ..    CMP   , ZERO
3.15    141202  ..    BE    P+ 002      <<=17>>
3.16    006400  ..    NOT   , NOP
3.17    017714  ..    BRE   P+ 014,I    <<=33->62>>
3.20    040012  @.    LOAD  P+ 012      <<=32>>
3.21    021002  ".    LDI   2
3.22    031004  2.    PCAL  C'DISPLAY'INIT
3.23    021006  ".    LDI   6
3.24    170010  ..    LRA   P+ 010      <<=34>>
3.25    140024  ..    BR    P+ 024      <<=51>>
3.26    000513  .K    DECX, MPYL
3.27    000535  .]    DECX, XAX
3.30    000540  .`    DECX, DEL
3.31    000471  .9    INCX, FIXT
3.32    001655  ..    DXCH, FNEG
3.33    000027  ..    NOP   , DTST
3.34    000005  ..    NOP   , DECX
3.35    013517  .0    TCBC  BIT 15
3.36    050105  PE    TBA   P+ 105      <<=143>>
3.37    047040  N     LOAD  DB+040,I,X
3.40    042522  ER    LOAD  P- 122,I    <<=177716>>
3.41    051117  RO    STOR  DB+117
3.42    051040  R     STOR  DB+040
3.43    047506  OF    LOAD  Q+ 106,I,X
3.44    020104   D    ----  << DOUBLE WORD >>
3.45    040524  AT    ????
3.46    040440  A     LOAD  P- 040      <<=6>>
```

Again, notice the 4 load instructions before the PCAL to DBOPEN at 3.10. The third load instruction at 3.6 shows a load of P+22. To the reside of the instruction <<=30>> is shown, indicating the location ferenced. Looking at the contents of 3.30 we notice the value 540. This needs to be changed to 542 using the modify command.

```
-M 3.30
  3.30    000540  .`    DECX, DEL                          :=542
  3.30    000542  .b    DECX, LDXB
-E

END OF PROGRAM

Finally we can run the program to verify that the patch is correct.

    :RUN XTEST;LIB=P;DEBUG


    *DEBUG* 0.0
    ?B 3.10
    ?R


    *BREAK* 3.10
    ?D S-3,4
    S-3       000513 000535 000542 000471
    ?D 542,I
    DB+542    +00005
    ?R

    Do you wish to extract only on-campus sections?
```

By looking at the contents of the stack we can observe the address for the mode parameter passed is indeed DB+542 and the contents of the address is 5.

Let's look at a second example. We have a program that uses a tape as input data and creates a report. We know the tape file is opened with the file name "CBM004". We would like to produce the same report using a disk file as input so we enter the appropriate file equation and run the program.

```
:FILE CBM004;DEV=DISC
:RUN XREPORT
```

After waiting a while we press the break key and type RECALL. We find that the program is still expecting to use a tape as input.

```
:RECALL
THE FOLLOWING REPLIES ARE PENDING:
?11:06/#S76/109/LDEV# FOR "CBM004" ON TAPE (NUM)?
```

We suspect the program has the bit set in the foptions for FOPEN which indicates to disallow file equations. To verify this we need to run the program again with the LMAP option to find what segments call FOPEN in the program.

```
:RUN XREPORT;LMAP

PROGRAM FILE XREPORT.MISC.ISIS

IIO'            PROG 0  16   0 SSL  0  34 205
DIO'            PROG 0  15   0 SSL  0  33 205
DATELINE        PROG 3  14   0 SSL  0  35 205
FSET            PROG 3  13   0 SSL  0  16 205
FOPEN           PROG 0  12   0 SSL  0   1   4
PRINTFILEINFO   PROG 3  11   0 SSL  0   2  46
SIO'            PROG 0  10   0 SSL  0  30 205
RIO'            PROG 0   7   0 SSL  0  32 205
TERMINATE'      PROG 0   6   0 SSL  0   2  41
FMTINIT'        PROG 0   5   0 SSL  0  21 205
TFORM'          PROG 0   4   0 SSL  0  22 205
OVFL'           PROG 0   3   0 SSL  0 143 203

301
```

We know that there is only 1 segment in the program and that FOPEN is at STT entry 12. Now we need to find where in the program FOPEN is called.

```
:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XREPORT
TYPE 'HELP' FOR ASSISTANCE.
```

Display the segment transfer table to find the last instruction in the segment.

```
-F 0.
  0.3015  116122  SEGMENT TRANSFER TABLE (PL-%16)  IIO'
  0.3016  115522  SEGMENT TRANSFER TABLE (PL-%15)  DIO'
  0.3017  116522  SEGMENT TRANSFER TABLE (PL-%14)  DATELINE
  0.3020  107122  SEGMENT TRANSFER TABLE (PL-%13)  FSET
  0.3021  100406  SEGMENT TRANSFER TABLE (PL-%12)  FOPEN
  0.3022  101047  SEGMENT TRANSFER TABLE (PL-%11)  PRINTFILEINFO
  0.3023  114122  SEGMENT TRANSFER TABLE (PL-%10)  SIO'
  0.3024  115122  SEGMENT TRANSFER TABLE (PL-%7)  RIO'
  0.3025  101042  SEGMENT TRANSFER TABLE (PL-%6)  TERMINATE'
  0.3026  110522  SEGMENT TRANSFER TABLE (PL-%5)  FMTINIT'
  0.3027  111122  SEGMENT TRANSFER TABLE (PL-%4)  TFORM'
  0.3030  161572  SEGMENT TRANSFER TABLE (PL-%3)  OVFL'
```

```
  0.3031  001403  SEGMENT TRANSFER TABLE (PL-%2)  SRS05S
  0.3032  000000  SEGMENT TRANSFER TABLE (PL-%1)
  0.3033  040016  STT LENGTH = %16
```

The last instruction will be 0.3014, so we look for a PCAL to FOPEN in 0.0 through 0.3014.

```
-F "PCAL  FOPEN",0/3014

  0.1574  031012  2.    PCAL  FOPEN
  0.1703  031012  2.    PCAL  FOPEN
-E

END OF PROGRAM
```

Now we run the program and use DEBUG to set breakpoints at the two calls to FOPEN.

```
:RUN XREPORT;DEBUG


*DEBUG* 0.1403
?B 1574,1703
?R


*BREAK* 0.1574
```

We know the parameters used in FOPEN by looking in the System Intrinsics manual. The layout of the stack before a call to FOPEN is as follows:

Location	Parameter for FOPEN
S-16	FORMALDESIGNATOR
S-15	FOPTIONS
S-14	AOPTIONS
S-13	RECORD SIZE
S-12	DEVICE
S-11	FORMS MESSAGE
S-10	USER LABELS
S-7	BLOCKING FACTOR
S-6	NUMBER OF BUFFERS
S-5	FILE SIZE (DOUBLE WORD)
S-3	NUMBER OF EXTENTS
S-2	INITIAL ALLOCATION
S-1	FILE CODE
S-0	OPTION VARIABLE MASK

We look now at the top 17 locations on the stack.

```
?D S-16,17
S-16      001416 002001 000000 177660 001426 000000 000000 000014
S-6       020040 020040 020040 020040 020040 020040 017440
```

Next we look at the file name used in the FOPEN. Since there are two calls to FOPEN we need to know if this is the open for the tape file or the report file.

```
?D 1416/2,10,A
DB+607    CBM004 .TAPE ...
```

This is indeed the FOPEN for the file CBM004. Let's look at the parameters for record size and device.

```
?=177660,I
 =-80
?D 1426/2,10,A
DB+613    TAPE .....
```

The file is opened for an 80 byte record and looking at the foptions value of 2001, we know bit 5 was indeed set to disallow file equations. Let's change the value of the foptions at S-15 to 1 to allow file equations then change the data stored at DB+613 and DB+614 to indicate a disk device rather than tape.

```
?M S-15
S-15      002001 :=1
?M 613,2
DB+613    052101 :="DI"
DB+614    050105 :="SC"
?D 613,2,A
DB+613    DISC
?R


*BREAK* 0.1703
```

This is the second FOPEN so resume execution.

```
?R


END OF PROGRAM

:
```

As we can see the program did not wait for a tape request so the disk file was used as input rather than the tape.

We would like to have the program allow file equations and default to disk rather than default to tape as it currently does. We start by running DECOMP to find where value 2001, the foptions value, is stored. We do not know the address using DEBUG since the foptions parameter is passed by value.

```
:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XREPORT
TYPE 'HELP' FOR ASSISTANCE.
```

First we decompile the statements close to the PCAL to FOPEN to see how the parameters are placed on the stack.

```
-1540
  0.1540  163001  ..    STD   DB+001,I
  0.1541  000647  ..    ZERO, FLT
```

```
0.1542  163000  ..    STD   DB+000,I
0.1543  140005  ..    BR    P+ 005       <<=1550>>
0.1544  041502  CB    LOAD  Q+ 102
0.1545  046460  M0    LOAD  P- 060,I,X <<=1465->53131>>
0.1546  030064  04    ----
0.1547  020016  .     MOVE  PB-DB SDEC=2
0.1550  034404  9.    LDPN  %4          <<=1544>>
0.1551  034403  9.    LDPN  %3          <<=1546>>
0.1552  140004  ..    BR    P+ 004      <<=1556>>
0.1553  052101  TA    MTBA  P+ 101      <<=1654>>
0.1554  050105  PE    TBA   P+ 105      <<=1661>>
0.1555  020000  .     MOVE  PB-DB SDEC=0
0.1556  040403  A.    LOAD  P- 003      <<=1553>>
0.1557  034403  9.    LDPN  %3          <<=1554>>
0.1560  000600  ..    ZERO, NOP
0.1561  171707  ..    LRA   S- 007
0.1562  010201  ..    LSL   1 BIT
0.1563  040016  @.    LOAD  P+ 016      <<=1601>>
0.1564  000600  ..    ZERO, NOP
0.1565  025120  *P    LDNI  80
0.1566  171707  ..    LRA   S- 007
0.1567  010201  ..    LSL   1 BIT
0.1570  000700  ..    DZRO, NOP
0.1571  021014  ".    LDI   12
0.1572  035006  :.    ADDS  %6
0.1573  040007  @.    LOAD  P+ 007      <<=1602>>
0.1574  031012  2.    PCAL  FOPEN
0.1575  051707  S.    STOR  S- 007
0.1576  035406  ;.    SUBS  %6
0.1577  051406  S.    STOR  Q+ 006
0.1600  140003  ..    BR    P+ 003      <<=1603>>
0.1601  002001  ..    ADD , DELB
0.1602  017440  .     TSBC  BIT 32,X
0.1603  141502  .B    BNE   P+ 002      <<=1605>>
0.1604  140042  ."    BR    P+ 042      <<=1646>>
0.1605  041406  C.    LOAD  Q+ 006
0.1606  031011  2.    PCAL  PRINTFILEINFO
0.1607  000707  ..    DZRO, DZRO
0.1610  021002  ".    LDI   2
0.1611  172003  ..    LRA   P+ 003,I    <<=1614->1645>>
0.1612  031005  2.    PCAL  FMTINIT'
0.1613  140020  ..    BR    P+ 020      <<=1633>>
```

We can see that the file name CBM004 is P-relative data and is loaded onto the stack by the LDPN instructions at 0.1550 through 0.1551. We can also see that the device type is loaded onto the stack by the LOAD and LDPN instructions at 0.1556 through 0.1557. The address of the file name is pushed onto the stack at 0.1561 then this word address is converted to a byte address by shifting the value left 1 bit at 0.1562. The value for the foptions is pushed onto the stack at the next instruction 0.1563, which is LOAD P+16. The value to the side of the instruction indicates the P-relative address of P+16 is 1601. By looking at the value at 0.1601 we see it is 2001. This value needs to be changed to the value one as we did using DEBUG.

```
-M 1601
  0.1601  002001  ..    ADD , DELB                          :=1
  0.1601  000001  ..    NOP , DELB
```

Finally, we need to change the value at 0.1553 through 0.1554 from "TAPE" to "DISC" so the file will default to the disk device.

```
-M 1553/1554
  0.1553  052101  TA    MTBA  P+ 101      <<=1654>>         :="DI"
  0.1553  042111  DI    LOAD  P+ 111,I    <<=1664->36270>>
  0.1554  050105  PE    TBA   P+ 105      <<=1661>>         :="SC"
  0.1554  051503  SC    STOR  Q+ 103
-D 1553/1554
  0.1553  042111 051503                                     DISC
-E
```

```
     END OF PROGRAM

Now we run the program using DEBUG to verify that our patch is correct.

     :RUN XREPORT;DEBUG

     *DEBUG* 0.1403
     ?B 1574
     ?R

     *BREAK* 0.1574

Looking at S-15, we see the foptions is now 1, and looking at DB+613
we see the contents is now "DISC".

     ?D S-16,17
     S-16      001416 000001 000000 177660 001426 000000 000000 000014
     S-6       020040 020040 020040 020040 020040 020040 017440

     ?D 1426/2,10,A
     DB+613    DISC ..........
     ?R

     END OF PROGRAM
```

The final example will show multiple solutions to solve a problem. We have a program which allows one to change the terminal type for $STDIN. The program does not need SM capability, yet it checks to see if the person running the program is MANAGER.SYS, and if not, the program terminates.

```
     :RUN XDEV


     USER:  MANAGER
     ACCOUNT:  ISIS
     GROUP:    MISC
     LOGON DEVICE:      23
     You must be MANAGER.SYS to run this program.

     END OF PROGRAM
```

A simple solution would be to write our own WHO subroutine which always returns the user as MANAGER.SYS and place it in an SL. The problem in doing this is that all programs which call the WHO intrinsic and use this SL will always return the user as MANAGER.SYS and we want only this one program to have MANAGER.SYS returned to it.

```
We start by looking at the program file information using
LISTDIR2.PUB.SYS.

     :RUN LISTDIR2.PUB.SYS

     LISTDIR2 C.01.00 (C) HEWLETT-PACKARD CO., 1977
     TYPE 'HELP' FOR AID

     >LISTF XDEV
     ********************
     FILE: XDEV.MISC.ISIS

     FCODE: PROG            FOPTIONS: STD,BINARY,FIXED
     BLK FACTOR: 1          CREATOR: **
     REC SIZE: 256(B)       LOCKWORD: **
     BLK SIZE: 128(W)       SECURITY--READ:    ANY
     EXT SIZE: 7(S)                 WRITE:   ANY
     # REC: 6                      APPEND:  ANY
```

```
# SEC: 7                          LOCK:     ANY
# EXT: 1                          EXECUTE: ANY
MAX REC: 6                      **SECURITY IS ON
MAX EXT: 1              COLD LOAD ID: %15115
# LABELS: 0             CREATED: MON, 18 JUL 1983
MAX LABELS: 0           MODIFIED: MON, 18 JUL 1983
DISC DEV #: 2           ACCESSED: MON, 18 JUL 1983
DISC TYPE: 0            LABEL ADR: **
DISC SUBTYPE: 8         SEC OFFSET: %1
CLASS: DISC             FLAGS: NO ACCESSORS
FCB VECTOR: %0

# SEG: 1                TOTAL DB: %25
STACK: %1440            DL: %0
MAXDATA: DEFAULT        CAP: IA,BA
>EXIT

END OF PROGRAM
```

We note that there is only one segment in the program. Next we run DECOMP to find where the WHO intrinsic is called and where the data for user, account, group and logon terminal is stored.

```
:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XDEV
TYPE 'HELP' FOR ASSISTANCE.

-F 0.
  0.360   114122  SEGMENT TRANSFER TABLE (PL-%13)  SIO'
  0.361   116122  SEGMENT TRANSFER TABLE (PL-%12)  IIO'
  0.362   101042  SEGMENT TRANSFER TABLE (PL-%11)  TERMINATE'
  0.363   110522  SEGMENT TRANSFER TABLE (PL-%10)  FMTINIT'
  0.364   111122  SEGMENT TRANSFER TABLE (PL-%7)  TFORM'
  0.365   122572  SEGMENT TRANSFER TABLE (PL-%6)  F'CONTRAP
  0.366   100442  SEGMENT TRANSFER TABLE (PL-%5)  TERMINATE
  0.367   100406  SEGMENT TRANSFER TABLE (PL-%4)  FOPEN
  0.370   114046  SEGMENT TRANSFER TABLE (PL-%3)  WHO
  0.371   103101  SEGMENT TRANSFER TABLE (PL-%2)  FCONTROL
  0.372   000021  SEGMENT TRANSFER TABLE (PL-%1)  TYPESET
  0.373   040013  STT LENGTH = %13

-F "PCAL  WHO",0/357


  0.47    031003  2.    PCAL  WHO
```

The WHO intrinsic is called from one place in the program, at 0.47. Now we need to find where the locations are placed on the stack in preparation for the call to WHO so we know where the data is to be stored.

```
  0.21    035007  :.    ADDS  %7        <-- Primary Entry TYPESET
  0.22    171700  ..    LRA   S- 000
  0.23    010201  ..    LSL   1 BIT
  0.24    051404  S.    STOR  Q+ 004
  0.25    035004  :.    ADDS  %4
  0.26    171700  ..    LRA   S- 000
  0.27    010201  ..    LSL   1 BIT
  0.30    051405  S.    STOR  Q+ 005
  0.31    035004  :.    ADDS  %4
  0.32    171700  ..    LRA   S- 000
```

```
0.33    010201  ..      LSL   1 BIT
0.34    051406  S.      STOR  Q+ 006
0.35    000706  ..      DZRO, ZERO
0.36    033405  7.      LLBL  TERMINATE
0.37    031006  2.      PCAL  F'CONTRAP
0.40    000706  ..      DZRO, ZERO
0.41    041406  C.      LOAD  Q+ 006
0.42    041405  C.      LOAD  Q+ 005
0.43    041404  C.      LOAD  Q+ 004
0.44    000600  ..      ZERO, NOP
0.45    171401  ..      LRA   Q+ 001
0.46    021035  ".      LDI   29
0.47    031003  2.      PCAL  WHO
```

We can see the parameters passed to the WHO intrinsic from the System Intrinsics manual. The parameters are placed on the stack as follows:

Location	Parameter for WHO
S-10	MODE
S-7	CAPABILITY
S-6	LOCAL ATTRIBUTES
S-5	USER NAME
S-4	GROUP NAME
S-3	ACCOUNT NAME
S-2	HOME GROUP
S-1	LOGON TERMINAL
S-0	OPTION VARIABLE MASK

Looking at 0.40 through 0.45 we see the loads to top of stack of the addresses for the parameters and at 0.46 the placing on top of stack of the option variable mask. The option variable mask is 35 octal or 29 decimal. The binary representation of 35 octal is 00/011/101. The bits which are set indicate which parameters, from right to left, that are passed to the intrinsic. The parameters for the WHO intrinsic are as follows:

WHO (mode,capability,lattr,usern, groupn,acctn,homen,term); O-V

Alligning the bits with the parameters of the intrinsic, we find the parameters passed are the fourth, fifth, sixth and eighth, which are USERN, GROUPN, ACCTN and TERM.

Looking at the loads to top of stack at 0.40 through 0.45, we see at 0.40 three words with the contents of zero loaded, which are the first three unreferenced parameters. At 0.41, we see a load of Q+6, which contains the byte address for the fourth parameter, USERN. At 0.42, we see a load of Q+5, which contains the byte address for the fifth parameter, GROUPN. At 0.43, we see a load of Q+4, which contains the byte address for the sixth parameter, ACCTN. At 0.44, a load of one word containing zero for the unreferenced seventh parameter. Finally, at 0.45, we see a load of the address of Q+1, which is the eighth parameter, TERMN.

We can run the program using DEBUG to verify this.

```
:RUN XDEV;DEBUG


*DEBUG* 0.21
?B 47
?R


*BREAK* 0.47
?DS-10,11
S-10      000000 000000 000000 000124 000114 000104 000000 000034
S+0       000035
?B 50
?R


*BREAK* 0.50
?D 124/2,4,A
DB+52     MANAGER
?D 114/2,4,A
DB+46     MISC
?D 104/2,4,A
DB+42     ISIS
?D 34,I
DB+34     +00023
?R


USER:  MANAGER
ACCOUNT:  ISIS
GROUP:    MISC
LOGON DEVICE:      23
You must be MANAGER.SYS to run this program.

END OF PROGRAM
```

Now that we know where the intrinsic WHO is being called and where the information is being stored, we can decide on a solution.

The first solution uses brute force. We decide to let the WHO intrinsic be called but will bypass the checking of the data for being the user MANAGER and the account SYS. This will allow any user to run this program. We use DECOMP to determine where the checking is done for the user to be MANAGER.SYS.

```
0.166   042105  DE    LOAD  P+ 105,I   <<=273->44674>>
0.167   053111  VI    STOR  DB+111,I
0.170   041505  CE    LOAD  Q+ 105
0.171   035040  :     ADDS  %40
0.172   040407  A.    LOAD  P- 007      <<=163>>
0.173   034407  9.    LDPN  %7          <<=164>>
0.174   034406  9.    LDPN  %6          <<=166>>
0.175   034405  9.    LDPN  %5          <<=170>>
0.176   021016  *.    LDI   14
0.177   171707  ..    LRA   S- 007
0.200   010201  ..    LSL   1 BIT
0.201   031013  2.    PCAL  SIO'
0.202   035407  ;.    SUBS  %7
0.203   171401  ..    LRA   Q+ 001
0.204   031012  2.    PCAL  IIO'
0.205   031007  2.    PCAL  TFORM'
0.206   041406  C.    LOAD  Q+ 006
```

```
0.207   140005  ..    BR     P+ 005      <<=214>>
0.210   046501  MA    LOAD   P- 101,I,X  <<=107->20147>>
0.211   047101  NA    LOAD   DB+101,I,X
0.212   043505  GE    LOAD   Q+ 105,I
0.213   051040  R     STOR   DB+040
0.214   170404  ..    LRA    P- 004      <<=210>>
0.215   010201  ..    LSL    1 BIT
0.216   021010  ".    LDI    8
0.217   020243  .     CMPB   PB-DB      SDEC=3
0.220   145503  .C    BNE    P+ 003,I    <<=223->236>>
0.221   041404  C.    LOAD   Q+ 004
0.222   140006  ..    BR     P+ 006      <<=230>>
0.223   000013  ..    NOP ,  MPYL
0.224   051531  SY    STOR   Q+ 131
0.225   051440  S     STOR   Q+ 040
0.226   020040        MVB    PB-DB SDEC=0
0.227   020040        MVB    PB-DB SDEC=0
0.230   170404  ..    LRA    P- 004      <<=224>>
0.231   010201  ..    LSL    1 BIT
0.232   021010  ".    LDI    8
0.233   020243  .     CMPB   PB-DB      SDEC=3
0.234   141502  .B    BNE    P+ 002      <<=236>>
0.235   140050  .(    BR     P+ 050      <<=305>>

0.236   000707  ..    DZRO,  DZRO
0.237   021002  ".    LDI    2
0.240   172003  ..    LRA    P+ 003,I    <<=243->304>>
0.241   031010  2.    PCAL   FMTINIT'
0.242   140030  ..    BR     P+ 030      <<=272>>
0.243   000041  .!    NOP ,  ZROB
0.244   054557  Yo    TBX    P- 157    <<=65>>
0.245   072440  u     ADDM   P- 040,I    <<=205->31214>>
0.246   066565  mu    CMPM   P- 165,I,X  <<=61->40464>>
0.247   071564  st    ADDM   Q+ 164
0.250   020142   b    MVLB   SDEC=2
0.251   062440  e     CMPM   P- 040,I    <<=211->47312>>
0.252   046501  MA    LOAD   P- 101,I,X  <<=151->21161>>
0.253   047101  NA    LOAD   DB+101,I,X
0.254   043505  GE    LOAD   Q+ 105,I
0.255   051056  R.    STOR   DB+056
0.256   051531  SY    STOR   Q+ 131
0.257   051440  S     STOR   Q+ 040
0.260   072157  to    ADDM   P+ 157,I    <<=437>>
0.261   020162   r    SCU    SDEC=2
0.262   072556  un    ADDM   P- 156,I    <<=104->41623>>
0.263   020164   t    SCU    SDEC=0
0.264   064151  hi    CMPM   P+ 151,X    <<=435>>
0.265   071440  s     ADDM   Q+ 040
0.266   070162  pr    ADDM   P+ 162      <<=450>>
0.267   067547  og    CMPM   Q+ 147,I,X
0.270   071141  ra    ADDM   DB+141
0.271   066456  m.    CMPM   P- 056,I,X  <<=213->51253>>
0.272   025426  +.    LDXN   22
0.273   044401  I.    LOAD   P- 001,X    <<=272>>
0.274   011202  ..    IXBZ   P+ 002      <<=276>>
0.275   140402  ..    BR     P- 002      <<=273>>
0.276   021054  ",    LDI    44
0.277   171726  ..    LRA    S- 026
0.300   010201  ..    LSL    1 BIT
0.301   031013  2.    PCAL   SIO'
0.302   035426  ;.    SUBS   %26
0.303   031007  2.    PCAL   TFORM'
0.304   031011  2.    PCAL   TERMINATE'
0.305   000606  ..    ZERO,  ZERO
0.306   021040  "     LDI    32
0.307   035014  :.    ADDS   %14
0.310   040004  @.    LOAD   P+ 004      <<=314>>
```

```
  0.311   031004  2.    PCAL  FOPEN
  0.312   051402  S.    STOR  Q+ 002
  0.313   140002  ..    BR    P+ 002      <<=315>>
  0.314   004000  ..    DEL , NOP
  0.315   141202  ..    BE    P+ 002      <<=317>>
  0.316   140037  ..    BR    P+ 037      <<=355>>
```

In instructions 0.166 through 0.205 we see the output of what the logon device is. At 0.206 the address of the USERN, Q+6, is loaded. Next, a branch to 0.214, to bypass the P-relative data "MANAGER ". Next, at 0.214, the address of the P-relative data is loaded onto the stack. Next, the address is logically shifted left 1 bit to form a byte address from the word address which was loaded. Next, a decimal 8, the number of bytes to compare, is loaded onto the stack. At 0.217, the CMPB instruction will compare 8 decimal bytes. At 0.220, the instruction says branch to 0.236 if the bytes compared are not equal. Therefore, if the contents of the data whose address is stored at Q+6 for a length of 8 bytes does not match "MANAGER " then branch to 0.236, otherwise continue. Next, at 0.221, the address of the ACCTN, Q+4, is loaded. The instructions at 0.222 branches around the P-relative data stored in 0.223 through 0.227. Next, at 0.230, the address of the P-relative data is loaded onto the stack. Next, the address is logically shifted left 1 bit to form a byte address from the word address. At 0.232, a decimal 8, the number of bytes to compare, is loaded onto the stack. At 0.233 is the compare bytes instruction, CMPB. At 0.234, the instruction is to branch to 0.236 if the 8 bytes are not equal, otherwise the next instruction at 0.235 is a branch to instruction 0.305.

Now we know the flow of logic for the comparison. If the USERN is not "MANAGER" then branch to 0.236. Next if ACCTN is "SYS" then branch to 0.305, otherwise branch to 0.236. To implement the "brute force" solution, instruction 0.206 is modified to BR P+77, which is a branch to 0.305, which completely bypasses any checking for MANAGER.SYS, and continues processing.

```
-M 206
  0.206   041406  C.    LOAD  Q+ 006                    :=140077
  0.206   140077  .?    BR    P+ 077      <<=305>>
-E

END OF PROGRAM
:RUN XDEV


USER:  MANAGER
ACCOUNT:  ISIS
GROUP:  MISC
LOGON DEVICE:  23
Change to term type: ?10

END OF PROGRAM
:
```

SOLUTION 2.

A second solution is to modify the P-relative data being checked. Since we are in the account "ISIS" we can change the account name being checked from "SYS" to "ISIS". The P-relative data is at 0.224 through 0.227.

```
-M 224/225
  0.224   051531  SY    STOR  Q+ 131                    :="IS"
  0.224   044523  IS    LOAD  P- 123,X    <<=101>>
  0.225   051440  S     STOR  Q+ 040                    :="IS"
  0.225   044523  IS    LOAD  P- 123,X    <<=102>>
-224
  0.224   044523  IS    LOAD  P- 123,X    <<=101>>
  0.225   044523  IS    LOAD  P- 123,X    <<=102>>
```

```
  0.226   020040          MVB   PB-DB SDEC=0
  0.227   020040          MVB   PB-DB SDEC=0
  0.230   170404   ..     LRA   P- 004      <<=224>>
 -E

END OF PROGRAM
:RUN XDEV


USER:  MANAGER
ACCOUNT:  ISIS
GROUP:    MISC
LOGON DEVICE!       23
Change to term type: ?10

END OF PROGRAM
:
```

There are some disadvantages to this method. The first is the user must be MANAGER.ISIS to run the program and cannot be any other user on the system as the patch in solution 1 allows. This may be what you want. The second disadvantage is the error message generated will need to be modified to state "You must be MANAGER.ISIS to run this program.", but as we now know, this can be easily done.

SOLUTION 3.

The third solution is a bit exotic, however this technique for patching may be necessary if the data for USERN, ACCTN, GROUPN and TERMN is used elsewhere and you wish them not to reflect what the WHO intrinsic would normally return, but rather other values. In this solution we will patch the program to show the USERN as "MANAGER", the ACCTN as "SYS" the GROUPN as "PUB" and the TERMN as 20 no matter who runs the program.

What we will do is increase the global-DB area by %14 words to allocate memory the program does not reference and place initial values in these locations. Then we will change the array pointers from pointing at there intended memory locations to instead point to our global memory locations we are allocating.

We know by looking at the output from LISTDIR2 that the total global-DB area is %25 words. Global-DB is initialized at run time from data within the object code. Word 2 of record 0 denotes the beginning record number of the image of the global-DB area of the stack. We need to allocate 4 words (8 bytes) of storage for USERN, 4 words for ACCTN and 4 words for GROUPN. What we will do is add 14 octal (12 decimal words) to the value in word 2 of record 0 so the loader will allocate not 25 octal words of global-DB storage but instead 41 octal words.

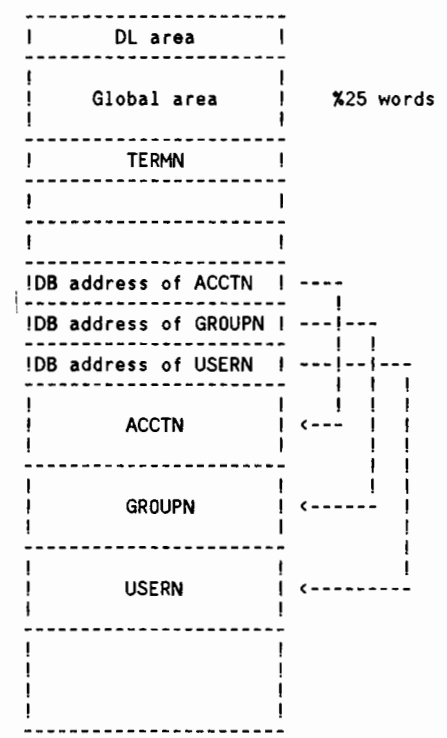
```
:RUN DISKED2.PUB.SYS

DISKED2 C.01.00 (C) HEWLETT-PACKARD CO., 1976
TYPE 'HELP' FOR INFO
>HELP

DISKED2 allows to dump and/or modify : file contents or
any disc sector (sys. mgr capability is required).

 B[ASE] [<ABS SEC #>]
 DEBUG
 DISC <LOG DEV #>
 D[UMP] [ [<REL SEC #>] [, <# OF SEC>] ] OR [<'ALL'>] [, A=ASCII ]
     (AT LEAST ONE PARAMETER MUST BE PRESENT.)
 F[ILE] <FILENAME>
 L[IST] [<DEVICE CLASS>] OR [<LOG DEV #>]
 M[ODIFY] <SEC NUM, REL WORD ADDR [,NUM OF WORDS]>
     (NEW VALUE STARTS WITH : # - DECIMAL, ' - ASCII)
```

```
 W[IDTH]
 E[XIT]

>FILE XDEV
>DUMP 1

LOGICAL SECTOR 1     *** BEGINNING OF DATA ***
SECTOR %00000220402        LDEV = %000004
000:  004600  000001  000025  000001  000002  001440  000000  177777
010:  000005  000000  000021  177777  000022  000004  000001  000001
020:  000000  000000  000000  000000  000000  000000  000000  000000
030:  000000  000000  000000  000000  000000  000374  000000  177774
040:  177774  177774  177774  177774  177774  177774  177774  177774
050:  177774  177774  177774  177774  177774  177774  177774  177774
060:  177774  177774  000000  066001  177777  000002  023401  177777


>M 1,2
LOGICAL SECTOR 1     *** BEGINNING OF DATA ***
SECTOR %00000220402        LDEV = %000004
002:  %000025,41
WRITTEN
>DUMP 1

LOGICAL SECTOR 1     *** BEGINNING OF DATA ***
SECTOR %00000220402        LDEV = %000004
000:  004600  000001  000041  000001  000002  001440  000000  177777
010:  000005  000000  000021  177777  000022  000004  000001  000001
020:  000000  000000  000000  000000  000000  000000  000000  000000
030:  000000  000000  000000  000000  000000  000374  000000  177774
040:  177774  177774  177774  177774  177774  177774  177774  177774
050:  177774  177774  177774  177774  177774  177774  177774  177774
060:  177774  177774  000000  066001  177777  000002  023401  177777
```

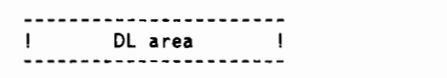
DB-relative data may be changed by modifying the image of the DB area in the object code. Knowing this, we now modify record 1 words %25 through %40 zero based to "MANAGER PUB SYS ". Note that the address for USERN is DB+25, the address for GROUPN is DB+31 and the address for ACCTN is DB+35.

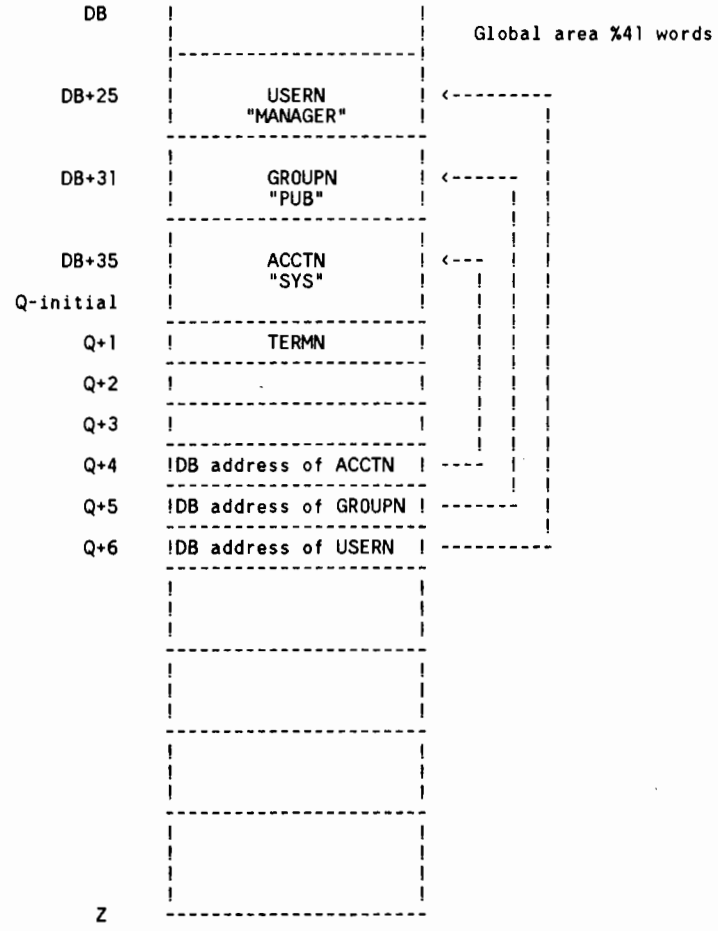
```
>DUMP 2

LOGICAL SECTOR 2
SECTOR %00000220403        LDEV = %000004
000:  000001  000000  000000  000000  000000  000000  000000  000000
010:  000000  000000  000000  000000  000000  000000  000000  000000
020:  000000  000000  003000  002400  177777  000000  000000  000000
030:  000000  000000  000000  000000  000000  000000  000000  000000
040:  000000  000000  000000  000000  000000  000000  000000  000000
050:  000000  000000  000000  000000  000000  000000  000000  000000
060:  000000  000000  000000  000000  000000  000000  000000  000000
070:  000000  000000  000000  000000  000000  000000  000000  000000


>M 2,%25,12
LOGICAL SECTOR 2
SECTOR %00000220403        LDEV = %000004
025:  %000000,'MA'
026:  %000000,'NA'
027:  %000000,'GE'
030:  %000000,'R '
031:  %000000,'PU'
032:  %000000,'B '
033:  %000000,'  '
034:  %000000,'  '
035:  %000000,'SY'
```

```
036:  %000000,'S '
037:  %000000,'  '
040:  %000000,'  '
WRITTEN
>EXIT

END OF PROGRAM

Running DECOMP, we can verify that our modification is correct.

:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XDEV
TYPE 'HELP' FOR ASSISTANCE.

-D DB+25
 DB+25   046501 047101 043505 051040 050125 041040 020040  MANAGER PUB
 DB+34   020040 051531 051440 020040 020040                   SYS
-EXIT
The contents of the run time stack for the original program is as
follows:
```




```
The contents of the run time stack for the patched program is to
be as follows:
```





Next we must find where the values of Q+4, Q+5 and Q+6 are set and patch the code to load the new DB+ values we have allocated instead of the ones generated by the compiler.

```
:RUN DECOMP.LIB.SYS

HP3000 DECOMPILER 6.1

FILE NAME? XDEV
TYPE 'HELP' FOR ASSISTANCE.

-
  0.21    035007  :.    ADDS  %7       <-- Primary Entry TYPESET
  0.22    171700  ..    LRA   S- 000
  0.23    010201  ..    LSL   1 BIT
  0.24    051404  S.    STOR  Q+ 004
  0.25    035004  :.    ADDS  %4
  0.26    171700  ..    LRA   S- 000
  0.27    010201  ..    LSL   1 BIT
  0.30    051405  S.    STOR  Q+ 005
  0.31    035004  :.    ADDS  %4
```

```
0.32    171700  ..    LRA   S- 000
0.33    010201  ..    LSL   1 BIT
0.34    051406  S.    STOR  Q+ 006
0.35    000706  ..    DZRO, ZERO
0.36    033405  7.    LLBL  TERMINATE
0.37    031006  2.    PCAL  F'CONTRAP
0.40    000706  ..    DZRO, ZERO
0.41    041406  C.    LOAD  Q+ 006
0.42    041405  C.    LOAD  Q+ 005
0.43    041404  C.    LOAD  Q+ 004
0.44    000600  ..    ZERO, NOP
0.45    171401  ..    LRA   Q+ 001
0.46    021035  ".    LDI   29
0.47    031003  2.    PCAL  WHO
```

Q+ relative memory is allocated when the program starts. By looking at instruction 0.21, we see 7 words allocated on top of stack. At 0.22 we see the address of S-0 stored at top of stack then logically shifted left 1 bit to form a byte address, then at 0.24 this value is stored at Q+4. Instructions 0.25 through 0.30 allocate 4 words and stores the byte address at Q+5. Instructions 0.31 through 0.34 similarly allocate 4 words and stores the byte address at Q+6. The instruction at 0.35 loads the final 3 words of array allocation onto the stack. Instruction 0.37 arms the CONTROL-Y trap to call TERMINATE if CONTROL-Y is pressed. As stated earlier, instructions 0.40 through 0.46 set up the parameters for the PCAL to the WHO intrinsic at 0.47.

Now that we know where the DB-relative addresses for the arrays are being set, we can patch the instructions for storing the starting addresses of the arrays in Q+4, Q+5 and Q+6 from the original addresses to the DB+ addresses we allocated.

```
-M 22
  0.22    171700  ..    LRA   S- 000          :=021035
  0.22    021035  ".    LDI   29
-M 26
  0.26    171700  ..    LRA   S- 000          :=021031
  0.26    021031  ".    LDI   25
-M 32
  0.32    171700  ..    LRA   S- 000          :=021025
  0.32    021025  ".    LDI   21
```

The instructions show LDI 29, LDI 25 and LDI 21. The numbers in these load immediate instructions are decimal. These numbers in octal are 35, 31 and 25 respectively, which can be seen by looking at the octal representation of the instruction.

We need to leave the adds to top of stack at 0.21, 0.25, 0.31 and 0.35 so the Q+ relative addressing will work. If we eliminated the adds, then some references could possibly be off by the number of words which were added to the stack if direct memory referencing is used.

Next we modify 0.40 to LDI 20 which loads a decimal 20 onto the stack, then instruction 0.41 to store the value in Q+1. Remember, the value of TERMN is stored at Q+1, so now the logon terminal is set to 20.

```
-M 40
  0.40    000706  ..    DZRO, ZERO           :=021024
  0.40    021024  ".    LDI   20
-M 41
  0.41    041406  C.    LOAD  Q+ 006         :=051401
  0.41    051401  S.    STOR  Q+ 001
```

Next we change instructions 0.42 through 0.47, the remainder of the instructions which set up the stack and then calls the WHO intrinsic, to NOP (no operation) instructions.

```
-M 42/47
  0.42    041405  C.    LOAD  Q+ 005                    :=0
  0.42    000000  ..    NOP , NOP
  0.43    041404  C.    LOAD  Q+ 004                    :=0
  0.43    000000  ..    NOP , NOP
  0.44    000600  ..    ZERO, NOP                       :=0
  0.44    000000  ..    NOP , NOP
  0.45    171401  ..    LRA   Q+ 001                    :=0
  0.45    000000  ..    NOP , NOP
  0.46    021035  ".    LDI   29                        :=0
  0.46    000000  ..    NOP , NOP
  0.47    031003  2.    PCAL  WHO                       :=0
  0.47    000000  ..    NOP , NOP
```

Looking at the code as patched we see:

```
-21

  0.21    035007  :.    ADDS  %7       <-- Primary Entry TYPESET
  0.22    021035  ".    LDI   29
  0.23    010201  ..    LSL   1 BIT
  0.24    051404  S.    STOR  Q+ 004
  0.25    035004  :.    ADDS  %4
  0.26    021031  ".    LDI   25
  0.27    010201  ..    LSL   1 BIT
  0.30    051405  S.    STOR  Q+ 005
  0.31    035004  :.    ADDS  %4
  0.32    021025  ".    LDI   21
  0.33    010201  ..    LSL   1 BIT
  0.34    051406  S.    STOR  Q+ 006
  0.35    000706  ..    DZRO, ZERO
  0.36    033405  7.    LLBL  TERMINATE
  0.37    031006  2.    PCAL  F'CONTRAP
  0.40    021024  ".    LDI   20
  0.41    051401  S.    STOR  Q+ 001
  0.42    000000  ..    NOP , NOP
  0.43    000000  ..    NOP , NOP
  0.44    000000  ..    NOP , NOP
  0.45    000000  ..'   NOP , NOP
  0.46    000000  ..    NOP , NOP
  0.47    000000  ..    NOP , NOP
-EXIT

END OF PROGRAM
```

Now we run the program.

```
:RUN XDEV


USER:  MANAGER
ACCOUNT:  SYS
GROUP:  PUB
LOGON DEVICE:  20
Change to term type: ?10

END OF PROGRAM
:
```

It should be clear now that object code is not sacred. The instructions in the object code can be decoded and modified as necessary to correct a problem or have the program perform differently than intended. Modification of instructions and P-relative data is quite simple. The insertion of P-relative data and instructions is not shown since it is quite a bit more involved, but is possible.

After patching several programs, one begins to better understand the internals of the

HP 3000 and appreciate the work of a compiler writer. Hopefully, one can now see that object code can be changed just as source code to better accommodate individual needs not considered when the source code was written.

PCAL TERMINATE

Title: Systematic Redesign: Modifying Object Code

Author: Phil Curry Coordinator of Administrative Computing Alvin Community College 3110 Mustang Road Alvin, Texas 77511

Phone: (713) 331-6111

Biographical Sketch:

I have been employed at Alvin Community College since 1976. I am responsible for administrative computing which includes our student records, payroll, personnel, and accounting systems. I have worked with the HP 3000 since 1977 starting with the Series II. I'm a member of the board of directors for the Greater Houston Regional User's Group and am a member of the Contributed Software Library committee. I have made numerous contributions to the CSL including a tape library system, Super Star Trek, and a program which allows one to reset the runonly option in a BASIC program.

SOFTWARE PROTOTYPING: TODAY'S APPROACH TO INFORMATION SYSTEMS DESIGN AND DEVELOPMENT

ORLAND LARSON
HEWLETT-PACKARD

Among the challenges facing the data processing community are the increasing costs and time associated with developing applications, the increasing backlog of applications, the excessive time spent maintaining applications, and the shortage of EDP professionals. In addition, systems implementation and functionality are impaired due to the lack of tools which involve end-users in the system development process.

Meeting these challenges requires a more progressive approach to applications development - one that is significantly different from traditional system development cycles. This approach is called SOFTWARE PROTOTYPING.

This paper defines software prototyping, identifies its major uses, reviews the step-by-step prototype development process, and discusses the resources and skills required to effectively prototype applications. It also addresses the problems and costs associated with software prototyping.

INTRODUCTION

The Changing Role of Data Processing

The data processing department has changed dramatically since the 1960's, when application development as well as production jobs were usually run in a batch environment with long turnaround times and out-of-date results.

The 1970's were a period of tremendous improvement for the data processing environment. One of the key developments of that period was the development and use of Data Base Management Systems (DBMS). This provided the basis for on line interactive applications. In addition, computers and operating systems provided programmers the capability of developing application pro- grams on line, sitting at a terminal and interactively developing, compiling, and testing these applications. The end user was also provided with easy to use on-line inquiry facilities to allow them to access and report on data residing in their data bases. This took some of the load off the programmers and allowed them to concentrate on more complex problems.

During the 1980's, for the Data Base Administrator and MIS manager, we see increased importance and use of centralized data dictionaries or "centralized repositories of information about the corporate data resources." We also see simpler and more powerful report writers for the end user and business professional. For the programmer, we see the use of very high level transaction processing languages to reduce the amount of code required to develop applications. Finally, the tools have been developed to effectively do software prototyping which will provide benefits to the end user as well as the application programmer and analyst.

Throughout the Seventies and Eighties, information has become more accurate, reliable, and available, and the end user or business professional is becoming more involved in the application development process.

Challenges Facing MIS

The MIS manager's number one problem is the shortage of EDP specialists. A recent Computerworld article predicted that by 1990 there will be 1/3 of a programmer available for each computer delivered in this country. Software costs are also increasing because people costs are going up and because of the shortage of skilled EDP specialists. The typical MIS manager is experiencing an average of two to five years of application backlog. This doesn't include the "invisible backlog", the needed applications which aren't even

requested because of the current known backlog. In addition, another problem facing MIS management is the limited centralized control of information resources.

The programmer/analyst is frustrated by the changeability of users' application requirements (the only thing constant in a user environment is change). A significant amount of programmers' time is spent changing and maintaining users' applications (as much as 60% of their time). Much of the code the programmer generates is the same type of routines such as error checking, formatting reports, reading files, checking error conditions, data validation, etc. This can become very monotonous or counter- productive for the programmer.

The end user or business professional is frustrated by the limited access to information needed to effectively do his/her day-to-day job. This is especially true for those users who know their company has spent a great deal of money on computer resources and haven't experienced the benefits. The user's business environment is changing dynamically and they feel MIS should keep up with these changes. MIS, on the other hand, is having a difficult time keeping up with these requests for application maintenance because of the backlog of applications and the shortage of EDP specialists. Once the user has "signed off" on an application, he is expected to live with it for awhile. He is frustrated when he requests what he thinks is a "simple change" and MIS takes weeks or months to make that change.

Traditional Approach to Application Development

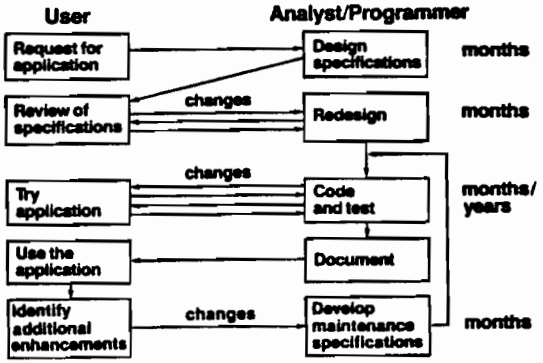
There are some myths concerning application development:

- Users know what they want
- Users can communicate their needs to MIS
- Users needs are static

The traditional approach to application development has serious limitations when applied to on-line, interactive information systems that are in a state of constant change and growth. Communications among the user, analyst, programmer, and manager tend to be imprecise, a detailed analysis prolongs the process to the annoyance of the user, and specifications are either ambiguous or too voluminous to read. To compound this problem, the user is often requested to "freeze" his requirements and subsequent attempts at change are resisted.

Let's review the traditional approach to application development.

TRADITIONAL APPROACH TO APPLICATION DEVELOPMENT



- The user first requests an application and then an analyst or programmer is assigned to the application.
- The analyst or programmer takes the oftentimes sketchy user specifications and designs more complete specifications.

- The user then reviews the analyst's interpretations of his specifications and probably makes additional changes.
- The analyst redesigns his specifications to adapt to these changes. (By this time, several days, weeks or months have gone by.)
- The user approves the specifications and a team of analysts and programmers are assigned to develop, test and document the application. (This may take months or years.)
- The user finally tries the application. Months or years may have gone by before the user gets his first look at the actual working application.

The question is: "Can MIS afford to continue using this traditional approach to application development?"

Prototyping Defined

According to Webster's Dictionary, the term prototype has three possible meanings:

1) It is an original or model on which something is patterned: an archetype.

2) A thing that exhibits the essential features of a later type.

3) A standard or typical example.

J. David Naumann and A. Milton Jenkins in a paper on software prototyping (see reference 3) believe that all three descriptions apply to systems development. Systems are developed as patterns or archetypes and are modified or enhanced for later distribution to multiple users. "A thing that exhibits the essential features of a later type" is the most appropriate definition because such prototypes are a first attempt at a design which generally is then extended and enhanced.

- The user, of course, will want additional changes or enhancements made to the application, to adjust the application to the "real world".
- Depending on the extent of these changes, additional maintenance specifications may have to be written and then coding, testing and documentation.
- The total application development process may take months or years and the maintenance of these applications may go on forever.

Software Prototypes

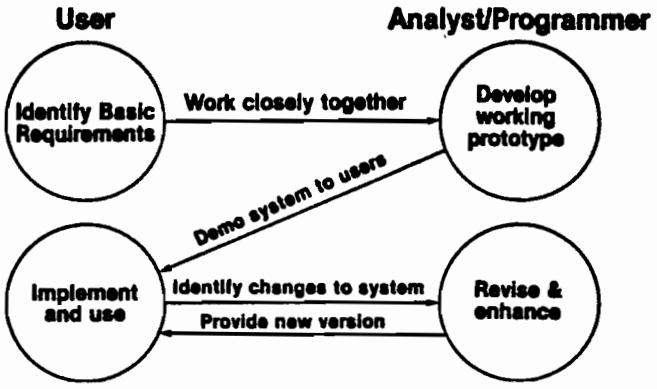
The process of software prototyping is a quick and relatively inexpensive process of developing and testing an application system. It involves the end user and programmer/analyst working closely to develop the application. It is a live, working system; it is not just an idea on paper. It performs actual work; it does not just simulate that work. It can be used to test out assumptions about users' requirements, system design, or perhaps even the logic of a program.

Prototyping is an iterative process. It begins with a simple prototype that performs only a few of the basic functions of a system. It is a trial and error process - build a version of the prototype, use it, evaluate it, then revise it or start over on a new version, and so on. Each version performs more of the desired functions and in an increasingly efficient manner. It may, in fact, become the actual production system. It is a technique that minimizes the dangers of a long formal analysis and increases the likelihood of a successful implementation.

The Prototype Model

Prototyping an information system can be viewed as a four step procedure.

PROTOTYPING APPROACH TO APPLICATION DEVELOPMENT



Step 1. Identify users' basic requirements:

- End user and programmer/analyst work closely together.
- Concentrate on users' most basic and essential requirements.
- Define data requirements, report formats, screens, and menus.
- Need not involve written specifications.
- For larger systems, a design team may need to spend a few weeks preparing a first-effort requirements document.

Step 2. Develop a working prototype:

- Programmer analyst takes the notes developed in the user discussions and quickly creates a working system.
- Designs and/or defines data base and loads subset of data.
- Makes use of defaults and standard report formats.
- Performs only the most important, identified functions.

Step 3. Implement and use the prototype:

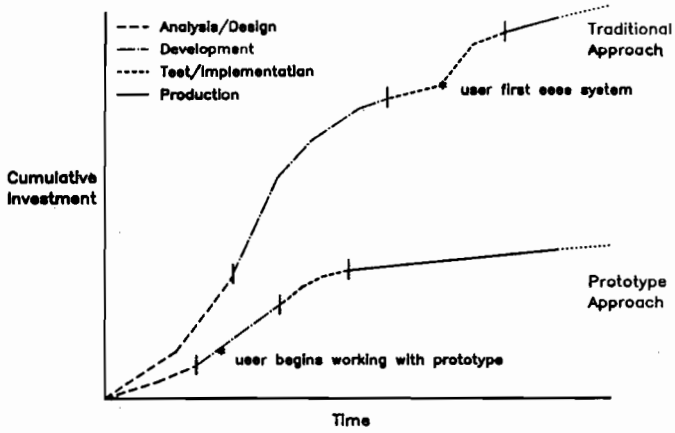
- Programmer/analyst demonstrates prototype to small group of users.
- Users may request enhancements during demo.
- Users make notes of all changes they would like made.

Step 4. Revise and enhance the prototype:

- Programmer/Analyst and user discuss desired changes.
- Changes and enhancements for the next version are prioritized.
- Programmer/Analyst creates next version.
- Go back to Step 3.

NOTE: Steps 3 and 4 are repeated until the system achieves the requirements of this small group of users. Then either introduce to a larger group of users for additional requirements or if enough users are satisfied, demo to management to gain approval for the production system.

PROTOTYPING VS TRADITIONAL APPROACH



Uses of Software Prototypes

1. To clarify user requirements:

   - Written specs are often incomplete, confusing, and take a static view of requirements.
   - It is difficult for an end user to visualize the eventual system, or to describe their current requirements.
   - It is easier to evaluate a prototype than written specifications.
   - Prototyping allows - even encourages users to change their minds.
   - It shortens the development cycle and eliminates most design errors.
   - It results in less enhancement maintenance and can be used to test out the effects of future changes and enhancements.

2. To verify the feasibility of design:

   - The performance of the application can be determined more easily.
   - The prototype can be used to verify results of a production system.
   - The prototype can be created on a minicomputer and then that software prototype may become the specifications for that application which may be developed on a larger mainframe computer.

3. To create a final system:

   - Part (or all) of the final version of the prototype may become the production version.
   - It is easier to make enhancements and some parts may be recoded in another language to improve efficiency or functionality.

Essential Resources

The following are the essential resources to effectively do software proto- typing:

1. Interactive Systems

   - Hardware and Operating System - When doing software prototyping, both the builder and the system must respond rapidly to the user's needs.

Batch systems do not permit interaction and revision at a human pace. Hardware and associated operating systems tailored to on-line interac- tive development are ideal for software prototyping.

2. Data Management Systems

- A Data Base Management System provides the tools for defining, creating, retrieving, manipulating, and controlling the information resources. Prototyping without a DBMS is inconceivable!

- A Data Dictionary provides standardization of data and file locations and definitions, a cross reference of application programs, and a built- in documentation capability. These are essential to managing the corporate resources and extremely useful when prototyping.

3. Generalized Input and Output Software

- Easy to use data entry, data editing, and screen formatting software are extremely helpful in the software prototyping process to allow the programmer to sit down at a terminal with a user and interactively create the user's screens or menus.

- Powerful easy-to-use report writer and query languages provide a quick and effective way of retrieving and reporting on data in the system. A report writer that uses default formats from very brief specifications is most useful in the initial prototype.

4. Very High Level Languages

- Traditional application development languages such as COBOL may not be well suited for software prototyping because of the amount of code that has to be written before the user sees any results.

- Very powerful high level (MACRO) languages that interface directly to a data dictionary for their data definitions are ideal. One statement in this high level language could realistically replace 20-50 COBOL state- ments. This reduces the amount of code a programmer has to write and maintain and speeds up the development process.

5. Library of Reusable Code

- A library of reusable code to reduce the amount of redundant code a programmer has to write is an important prototyping resource.

- This code could represent commonly used routines made available to programmers.

Potential Problems

What are the problems with prototyping? How can data processing management control its use and keep it within bounds?

One problem with prototyping is the acceptance of this method by the systems people. It also may encourage the glossing over of the systems analysis portion of a project. It may be difficult to plan the resources to develop a system. Programmers may become bored after the nth iteration of the proto- type. Testing may not be as thorough as desired and it might be difficult to keep documentation on the application up to date because it is so easy to change.

Even with these concerns, prototyping provides a very productive user-designer working relationship. So it behooves all data processing executives to learn to use this powerful tool creatively and to manage it effectively.

The advantages of prototyping greatly outweigh the problems.

Cost and Efficiency

It has been found that there is an order of magnitude decrease in both develop- ment cost and time with the prototype model.

It is often difficult to estimate the cost of an application system because the total costs of development, including maintenance are usually lumped together. The cost of implementing the initial system is much lower than the traditional approach (typically less than 25%).

However, software prototyping could be expensive in three ways:

1. It requires the use of advanced hardware and software.
2. It requires the time of high level users and experienced designers.
3. Efficiency may be compromised.

The main thing to remember is that the main focus of prototyping is not so much efficiency but effectiveness.

Summary

Prototyping is truly a "state of the art" way of developing applications.

- Software prototyping promotes an interactive dialogue between the users and the programmer, which results in a system being developed more quickly, and results in an interactive development approach which is friendlier for the end user.

- The prototype provides a live working system for the users to experiment with instead of looking at lengthy specifications.

- The users are provided with an early visualization of the system which allows them to immediately use it.

- The users are allowed and even encouraged to change their minds about user interfaces and reports.

- Maintenance is viewed right from the beginning as a continuous process and because the prototype is usually written in a very high level lan- guage, changes are faster to locate and easier to make.

- Software prototyping results in:

* Users who are much more satisfied and involved in the development process.

* Systems that meet the user's requirements and are much more effective and useful.

* Improved productivity for all those involved in software prototyping: the users, the analysts, and the programmers.

Hewlett-Packard's Prototyping Tools

Hewlett-Packard is one of the few vendors that supplies the majority of the tools needed to effectively do software prototyping.

* Interactive Systems

- HP3000 (All Series) - MPE Operating System

* Data Management Systems

- IMAGE/3000 - KSAM/3000 - MPE files - DICTIONARY/3000

* Generalized Input/Output Software

- VPLUS/3000 - QUERY/3000 - REPORT/3000 - INFORM/3000 - DSG/3000

* Very High Level Languages

- TRANSACT/3000

Canning, Richard G., "Developing Systems By Prototyping," EDP Analyzer (19:9) Canning Publications, Inc., September, 1981.

Naumann, Justus D. and Jenkins, A. Milton, "Prototyping: The New Paradigm for Systems Development," MIS Quarterly, Vol. 6, No. 3, September 1982.

Naumann, Justus D., and Galletta, Dennis F., "Annotated Bibliography of Proto- typing for Information Systems Development," Management Information Systems Research Center Working Paper (MISRC-WP-82-12), September 1982.

Note: The above working paper as well as the paper by Naumann and Jenkins entitled "Prototyping: The New Paradigm for Systems Development," MIS Research Center-Working Paper (MISRC-WP-82-03), October 1981, are available for $3.00 each from :

University of Minnesota Systems Research Center School of Management 269 19th Avenue South University of Minnesota Minneapolis, Minnesota 55455

or by calling 612-373-7822.

Podolsky, Joseph L., "Horace Builds a Cycle," Datamation, November 1977, pp.162-186.

DATA BASE DESIGN TOOLS:

A Survey of Current Research, an Opportunity for Customer Input

Abe Lederman
Hewlett-Packard
Information Networks Division
Cupertino, California

ABSTRACT:

Designing a data base can be a time consuming, tedious, and complex task. The ad-hoc methods used to design a data base often lead to data bases that are poorly designed and which adversely affect overall system performance.

In this paper we identify and briefly describe tools that can assist with the different phases of data base design, introducing methodology to the process and automating some of the more tedious aspects of data base design.

The purpose of this paper, and the accompanying presentation is to introduce our audience to current data base design tools research and give them an opportunity to give us at Hewlett-Packard some feedback as to the usefulness of such tools. A survey on data base design will be distributed at the presentation.

In writing this paper, the author has drawn on work that has conducted at Hewlett-Packard part of an investigation into data base design tools. At this time Hewlett-Packard has not made any commitment to actually develop any of the tools described in this paper

1. INTRODUCTION

Designing a data base can be a time consuming, tedious, and complex task. This is especially true when large data bases, consisting of hundreds of items and many data sets, are designed. In most cases data bases are designed in an ad-hoc trial-and-error manner without the benefit of methodologies or tools to assist in the design process.

The ad-hoc methods used to design a data base often lead to data bases that are poorly designed. A poor data base design will likely result in an information system with poor performance. In addition a poor data base design will make it harder to develop the application software to operate on information in the data base. Once a poorly designed data base is up and running, it can be very costly to restructure the data base to correct the design errors that should have been detected during the design phase.

For the past several months I have been involved in an investigation at Hewlett-Packard to try and determine what kinds of tools Hewlett-Packard can develop to help our customers design both network (IMAGE) and relational data bases. In this paper I am going to share with you some of what I have learned in this investigation.

This paper divides the data base design process into six phases. For each phase I identify and briefly describe tools that can assist with the task of data base design. It is not possible in a paper this brief to go into as much detail as is necessary to do justice to the material presented. The goal of this paper is to introduce the reader to data base design tools research. A select number of papers addressing different areas of data base design tools research are referenced throughout the paper.

I feel, and will try to show in this paper, that the use of data base design tools can :

- **Improve design quality**
- **Shorten design cycle**
- **Improve data base performance**

We are interested in learning how you currently design your databases, the difficulties you encounter, and what tools (other than those described in this paper) will help you be more productive and more successful in your data base design activities. At the conference we will distribute a survey on data base design which we strongly encourage you to fill out and return to Hewlett-Packard.

One thing to keep in mind in reading through this paper is that I anticipate that some of the data base design tools described in this paper will be mostly useful in projects involving the design of large data bases.

2. DESIGN PHASES

I have chosen to divide the data base design process into six phases :

REQUIREMENT SPECIFICATION - Information-oriented and processing-oriented requirements are collected and analyzed.

VIEW MODELLING - Local Entity-Relationship models are constructed from the information-oriented requirements. The Entity-Relationship (E-R) model is a conceptual organization of the data which allows the user to simply think of data objects (entities) and the relationships between them. The E-R model is described in section 2.2.1.

VIEW INTEGRATION - Local E-R models constructed in the previous phase are integrated into one consistent global conceptual schema.

SCHEMA MAPPING - Global conceptual schema is translated into a normalized logical schema for the target DBMS.

PHYSICAL DESIGN - Transaction models are constructed for the most active transactions. These models are input to physical design tools which assist with such design decisions as record segmentation, record joining, and the selection of primary and secondary indices.

DESIGN EVALUATION - Data base performance is estimated through the use of analytical modelling tools and/or data base prototyping tools.

The process of data base design is iterative in nature, both within a given phase, and among the different phases. For example, an inconsistency in the view integration phase may require a change to a local E-R model developed in the view modelling phase. This in turn may require modifying some requirements specified in the requirement specification phase. Similarly, the physical design and design evaluation phases form an iterative loop. A physical design change will require the design to be re-evaluated, which in turn may require additional changes to the physical design, and so on. Data base design tools must facilitate this iterative nature of the data base design process.

Let us now take a look at what kinds of tools can be developed to assist in each of these phases of the data base design process.

2.1 Requirement Specification

The requirement specification phase provides the initial input to the other phases of the design process. This input as proposed by Kahn [Kahn] should consist of both information-oriented and processing-oriented requirements. Information-oriented requirements describe the structure of the information that one is trying to model, and are used to construct the local E-R models in the view modelling phase. Processing-oriented requirements describe the transactions that will be performed against the data base, and are used to construct transaction models (see section 2.5.1) in the physical design phase.

Requirement specification tools should facilitate the input, edit, output, and verification of requirements. These tools should also enable designers to document the data base requirements and generate requirement analysis reports in a form that can be reviewed by end-users. Further

investigation is required to determine whether tools should be provided to assist the designer in mapping requirement specifications into E-R models and transaction models. Alternatively, the initial input of requirements to the data base design tools might occur in the view modelling and transaction modelling phases.

Data base requirement specification may be viewed as a subset of gathering requirements for the information system as a whole. Although a great deal of research [IEEE] into high level languages for requirement specification and analysis of information systems has been conducted, the specific problem of data base requirement specification has not been addressed to any significant extent. One possibility is to provide tools to automatically extract data base requirements from information system requirements.

2.2 View Modelling

In the view modelling phase of the data base design process we take the information-oriented requirements collected in the previous phase and represent them in a form that can be manipulated by the design tools. In this section I describe an E-R model that is used to model the information-oriented requirements.

The design of a large centralized data base requires the input of many users. One way in which this task can be simplified is by independently specifying the requirements for the different functions of an organization. Each such set of requirements is called a user view. For example, user views might be specified for the payroll, personnel, inventory and accounting functions of a company. From each user view independent local E-R models are developed.

The number of user views required to model a data base is dependent on the complexity of the data base design, and on the number of transactions that are encompassed by each user view. At one extreme [Hubbard, Yao], a local E-R model is constructed for each transaction against the data base. At the other extreme, the system analyst or DBA manually performs the task of view integration and only constructs one E-R model for the entire data base. A more common approach is for a view to encompass multiple transactions, which are typically the transactions that a user is responsible for, or the transactions for a logical subdivision of an enterprise. This approach, of modelling the information structure for a set of transactions with one E-R model, requires less input during the modelling phase, and results in less problems during the view integration phase, since a smaller number of views need to be integrated.

2.2.1 The E-R model

The Entity-Relationship-Attribute (E-R) model developed by Chen [Chen] provides a fairly natural way for modelling the information-oriented requirements. The E-R model, with various extensions, has been used in much of the research into computer-assisted data base design tools [Batini].

The E-R model consists of entities, attributes, and relationships among entities. An entity is any identifiable "thing". Entities of the same type are grouped into an entity-set. EMPLOYEE, DEPARTMENT, and CAR are some examples of entities.

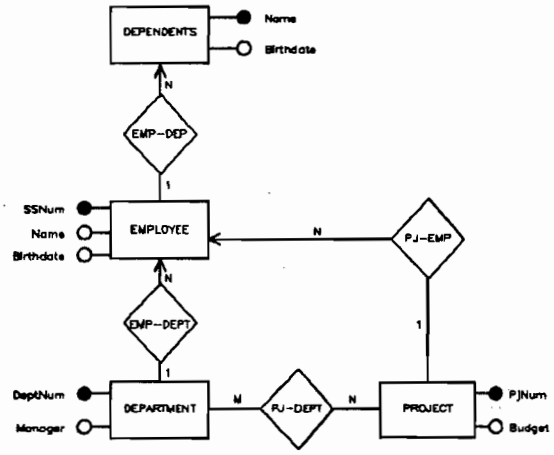
Entities have properties, called attributes. For example, the EMPLOYEE entity may have the following attributes : name, social security number, address, and birth-date. An attribute or set of attributes that uniquely identify an entity in an entity-set is called a key.

A relationship expresses an association among two (typically) or more entities. PJ-EMP is an example of a relationship between EMPLOYEE and PROJECT. Relationships can be classified into three categories : one-to-one (1:1), one-to-many (1:N), and many-to-many (M:N).

A graphical technique, the Entity-Relationship Diagram (ERD), may be employed to visually depict entities and relationships. In an ERD :

1. Entity sets are represented by a rectangular box.
2. Attributes are represented by circles.
3. Key attributes are represented by filled-in circles.
4. Relationships are represented by a diamond-shaped box. The relationship type (1:1, 1:N, M:N) is labelled along the lines connecting the entity boxes to the relationship diamond. For 1:N relationships an arrow points from the 1-entity to the N-entity.

The diagram below is an example of a very simple ERD.



There are two main reasons why the E-R model (as opposed to a network or relational model) is used to model the local views. First, at this early stage in the design process, it is important that the design be DBMS independent. The designer should defer the decision of selecting a target DBMS until the schema mapping phase. The E-R model provides a neutral model that is natural to use, straightforward to translate into a relational schema, and only a bit trickier to translate into a network schema. Second, the E-R model permits more of the semantics of the information system being developed to be captured than is possible with the network or relational model.

Graphic-oriented tools, similar to CAD (Computer Aided Design) tools used in various engineering disciplines, might provide a suitable interface for de eloping the E-R models. A graphic-oriented interface might allow system analysts and end-users to work together in developing their E-R models.

2.3 View Integration

The objective of the view integration phase is the merging of the local E-R models, developed in the previous phase, into a single, consistent global schema. This phase is obviously not required if only one view was created in the view modelling phase.

The first step in view integration involves identifying and resolving inconsistencies and redundancies among the local E-R models. Although design tools will not actually resolve inconsistencies or redundancies, they can assist the DBA or system analyst by identifying potential integration problems.

Some examples of inconsistencies that might occur include :

- SYNONYMS - The use of different names for the same object (entity, relationship or attribute).
- HOMONYMS - Using the same name for different objects.
- Type/size inconsistencies.
- Relationship inconsistencies. For example, defining a relationship between two entities as 1:N in one instance and as M:N in another instance.
- An entity is defined as an attribute of another entity.

Two examples of redundancies that might occur include :

- Redundant relationships. For example, A (an entity) is related 1:1 to B, B is related 1:1 to C, and A is related 1:1 to C. This last relationship is not necessary since it can be derived from the other two relationships.
- Redundant attributes. Attributes that appear in more than one entity. For example, NAME is defined as an attribute of both EMPLOYEE and EMPLOYEE-BENEFITS (with SS-NO as the key).

Once all the inconsistencies and redundancies have been resolved, the next step is to actually merge the local E-R models into the global conceptual schema. A design tool can perform this task, although the designer may wish to control how the merging is performed where alternative ways of merging exist.

2.4 Schema Mapping

In this phase, the global conceptual schema is mapped into a schema for the target DBMS. This phase might be viewed as part of the physical design phase. Physical design decisions such as index selection, however, are deferred until the physical design phase. In this phase we are mapping from the E-R model, expressing the global conceptual schema, into a "vanilla" schema of the target DBMS.

Mapping into a relational model is straightforward, while mapping into a network model may be somewhat trickier Tools can be provided to perform the schema mapping. As with view integration, alternative mappings may exist, requiring the designer to interact with the mapping tool in choosing among the different alternatives.

As part of the process of schema mapping, we want to generate a schema that is normalized. The algorithms to generate a normalized schema may be incorporated into the schema mapping tool. I now briefly describe normalization, an important concept in data base design.

2.4.1 Normalization

Normal forms [Kent] were first defined for relational data bases, although they are also applicable to the design of network data bases such as IMAGE. The normal forms can be viewed as guidelines to be followed in designing a data base in order to avoid update problems resulting from redundant data. These guidelines are for the most part common sense rules that most data base designers follow, perhaps without knowing that they are performing normalization.

Take as an example the SUPPLY table below. The key is the composite (S#, PART#). CITY, however, is only dependent on S#.

S#	PART#	QTY	CITY
S1	P1	300	New York
S1	P2	200	New York
S1	P3	400	New York
S2	P2	250	Palo Alto
S2	P4	500	Palo Alto
S3	P1	300	Sunnyvale

The above table design has several problems :

- CITY is repeated for every record having a PART# supplied by the same S#, resulting in wasted storage.
- If CITY changes for a given S#, then every record specifying a PART# supplied by the same supplier must be updated. In addition to the cost of doing all these updates, inconsistencies might arise with different records showing different cities for the same supplier.
- If at some point there are no parts supplied by a supplier, then there is no record in which to store the supplier's CITY.

The solution, obvious enough, is to normalize the above table by decomposing it into the two normalized tables shown below.

S#	PART#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P2	250
S2	P4	500
S3	P1	300

S#	CITY
S1	New York
S2	Palo Alto
S3	Sunnyvale

It should be noted, however, that it is not always desirable to strictly follow the normalization guidelines. There may be times where for performance reasons, it may be advantageous to maintain redundant data in a table. The normalized schema generated by the schema mapping tool is input to physical design tools where controlled denormalization is done when performance tradeoffs require it.

2.5 Physical Design

In the physical data base design phase, we take the "vanilla" schema generated in the schema mapping phase together with a description of the most active transactions that will be running against the data base and design the physical structure and access paths to the data base that will optimize the performance of the specified transactions.

This task, although it sounds simple, is in practice very complex. The designer must be able to evaluate the storage/CPU usage tradeoffs as well as tradeoffs in the performance of different transactions of many alternative data base designs.

The design evaluation phase which follows physical design in this paper is not really separate from the physical design phase. We have separated the two phases, since a separate set of tools can be identified to assist in the two phases. The physical design tools, however, will have to invoke some of the design evaluation tools in order to be able to optimize data base performance.

2.5.1 Transaction Modelling

In order for the physical design tools to be able to assist with producing an optimized physical design, they must be provided with a description of the workload that is going to be performed against the data base. In most cases the 80/20 rule applies. This rule states that the 20% most active transactions account for 80% of the workload on the data base. Therefore, it is only necessary to describe a few of the most active transactions to the physical design tools to provide a fairly accurate description of the data base workload.

The transaction models are constructed from the processing-oriented requirements collected during the initial requirement specification phase. I have, however, deferred discussing the transaction modelling step until now because constructing these models requires thinking in terms of physical data base structures. Physical design considerations should be avoided until the physical design phase.

The table below shows a transaction model for a report that outputs for every project is overbudget a list of all employees on the project and their departments. This transaction model is similar to models proposed in [Ceri, Teorey].

OBJECT	ACCESS MODE	RECORDS PROCESSED	RECORDS SELECTED	OPERATION	LINK USED NEXT
PROJECT	SERIAL	100	15	GET	PJ-EMP
PJ-EMP	INDEX	10	10	GET	EMPLOYEE
EMPLOYEE	DIRECT	1	1	GET	EMP-DEPT
EMP-DEPT	INDEX	1	1	GET	

FREQ: 2/MONTH WEIGHT: 1 MODE: BATCH

For each transaction we specify (at bottom of table):

FREQ - Frequency of execution.

WEIGHT - Weight factor for frequency. Typically 1. As an example, we might assign a weight factor greater than 1 to a transaction that is executed on an infrequent basis, but requires fast response time.

MODE - Batch or Online. Batch transactions are given a somewhat lower weight factor than online transactions.

For each access operation in the transaction we specify:

OBJECT - The object to be accessed, either an entity or a relationship.

ACCESS MODE - Serial, direct (hashed), or indexed. This is only a hint to the design tools. When considering overall data base performance, an indexed access, for example, may have to be converted into a serial access if it is not cost-effective to provide the required index.

RECORDS PROCESSED - Estimate of the average number of records that will have to be processed.

RECORDS SELECTED - Estimate of the average number of records that will be selected from those processed. In the above example, we estimate that 15 of the 100 project records accessed will be overbudget, requiring access of the PJ-EMP, EMPLOYEE, and EMP-DEPT records.

OPERATION - Get, put, update, or delete.

LINK USED NEXT - Object to be accessed following current access.

The above transaction model is only a first attempt at developing a model that can be used for describing transactions to the design tools. There are still some problems with this model that need to be resolved. One major problem is that the model needs to be able to describe the data that the

transaction is going to operate on in as physical-structure independent a manner as possible in order to give the physical design tools the freedom to do their job.

Perhaps another approach to this problem, which deserves further investigation, is the use of a high-level non-procedural language such as a relational DBMS query language as the language to model transactions in.

2.5.2 Physical Design Assistance

Some of the physical design decisions that tools can assist with include :

- **Record segmentation.** Since some transactions access just a subset of a record, it might make sense to segment the record along such usage lines, improving the performance of the transactions that only need to access the record subset. Smaller record sizes reduce the number of I/Os performed (assuming that we access the records in the order they are clustered), since it is possible to block more records together. Also smaller record sizes result in less wasted storage.

- **Record joining.** This is the opposite of record segmentation and is done to improve the performance of high frequency transactions which access data from two or more records together. The tradeoff to consider is that the resulting denormalized design may lead to redundancy problems.

- **Data base partitioning.** For performance reasons, maintainability reasons, or because of limitations such as the 256 item limit in IMAGE, it might be desirable to partition a data base into two or more data bases. Design tools can suggest how to best partition the data base.

- **Index selection.** Design tools can be built to assist in selecting primary and secondary indices based on access paths needed to run the specified transactions. Tradeoffs have to be evaluated between improving the performance of the transaction accessing the data base through the index, worsening the performance of the transactions that have to update the index, and the increase in storage requirements from the added index.

- **Sort item selection.** Design tools can assist in evaluating tradeoffs involving the addition of a sort item to a data set.

- **Prime capacities.** It has been shown that hashing for master data sets in IMAGE improves if prime capacities are used. Prime capacities, slightly higher than required capacity, can be automatically assigned by a design tool.

- **Blocking factors.** Optimal blocking factors can be calculated for each data set, relieving the designer from the task.

There is a great deal of literature describing algorithms used to optimize physical data base design parameters. A good survey paper is [Chen2].

2.6 Design Evaluation

Physical data base design involves making a lot of design decisions. The effects of such decisions on data base performance are not always obvious. There is a need for VISICALC like tools that can

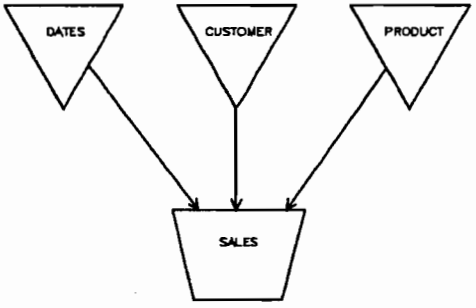
answer "what if" questions about data base performance. These tools need to be able to predict performance given a description of the data base structure and a description of the workload that is going to be run on the data base.

The tools that can be used to evaluate data base performance fall into two classes : analytical modelling tools and data base prototyping tools.

2.6.1 Analytical Modelling Tools

Making some simplifying assumptions, we can calculate the cost of running a transaction in terms of the average number of I/Os required to execute the transaction, ignoring all other costs. A design evaluation tool can be developed that will calculate the total I/O cost for a specified transaction workload. The following example illustrates the calculations that would be performed by such a tool in evaluating a very simple IMAGE design in which two transactions are executed.

The IMAGE design we are evaluating is a simplified version of the STORE data base in the IMAGE manual. This data base consists of two manual masters, one detail data set, and an optional auto master which provides a search path on PURCHASE-DATE. The goal of this evaluation is to determine whether the search path on PURCHASE-DATE should be provided or not.



There are two transactions performed against the STORE data base. The first transaction, SALES-POSTING, is an online transaction, executed 200 times per day. Each time the transaction is executed an average of 5 entries are posted to the SALES data set. A transaction model for this transaction is shown below :

SALES-POSTING

OBJECT	ACCESS MODE	RECORDS PROCESSED	RECORDS SELECTED	OPERATION	LINK USED NEXT
SALES		5		PUT	

FREQ: 200/DAY WEIGHT: 1 MODE: ONLINE

The second transaction, SALES-REPORT, is a batch transaction, executed once a day, which generates a report of sales information, customer information and product information for all sales with a specified PURCHASE-DATE. The transaction model for this transaction is shown below :

SALES-REPORT

OBJECT	ACCESS MODE	RECORDS PROCESSED	RECORDS SELECTED	OPERATION	LINK USED NEXT
SALES	SERIAL	20,000	1,000	GET	CUSTOMER
CUSTOMER	DIRECT	1	1	GET	PRODUCT
PRODUCT	DIRECT	1	1	GET	

FREQ: 1/DAY WEIGHT: 0.8 MODE: BATCH

The following assumptions have been made about the number of I/Os performed by each IMAGE intrinsic call:

	I/Os	
DBGET (master-calculated)	1.7	Assumes that we have to follow some synonym chains.
DBGET (detail-serial)	0.2	Assumes 5 records per block.
DBGET (detail-chain)	1	Does not assume that records are loaded in chain sequence.
DBPUT (detail)	2 + 4.7P	P is the number of search paths to the detail data set. 1 I/O - read a block with free space 1 I/O - write the block to disc 1.7P I/O - read the master record 1P I/O - update the master record 2P I/O - read and update previous end of chain

Let us now calculate the number of I/Os performed by the SALES-POSTING transaction when we do not have a search path on PURCHASE-DATE, and when we do have one.

The SALES-POSTING transaction calls the DBPUT intrinsic five times. Using the estimate :

$$I/Os = 2 + 4.7P$$

with P=2 (search paths on CUSTOMER and PRODUCT), each DBPUT intrinsic will perform 11.4 I/Os, and the SALES-POSTING transaction will perform a total of 57 I/Os.

with P=3 (search path on PURCHASE-DATE added), each DBPUT intrinsic will now perform 16.1 I/Os, and the SALES-POSTING transaction will perform a total of 80.5 I/Os.

Let us now do the same calculations for the SALES-REPORT transaction. Without a search path on PURCHASE-DATE, the transaction will call the DBGET intrinsic 20,000 times. For each of the 1,000 records we estimate will match the specified PURCHASE-DATE, a DBGET call to access the CUSTOMER data set and a DBGET call to access the PRODUCT data set will have to be made. The total number of I/Os performed by the transaction is as follows :

20,000 DBGETs (detail-serial) * 0.2	=	4,000
1,000 DBGETs (master-calculated) * 1.7	=	1,700
1,000 DBGETs (master-calculated) * 1.7	=	1,700
TOTAL	=	7,400

If we add a search item on PURCHASE-DATE, the 20,000 calls to DBGET in serial mode, are replaced by 1,000 calls to DBGET in chain mode, each call performing one I/O. Adding the search item on PURCHASE-DATE saves us 3,000 I/Os (4,000 - 1,000) over doing the serial DBGETs, and the total number of I/Os performed by the SALES-REPORT transaction is now only 4,400 I/Os.

The table below summarizes the remaining calculations that are necessary to determine which of our alternative designs will require the least number of total I/Os for the entire transaction workload.

TRANSACTION	WEIGHTED FREQ (tran/day)	DESIGN #1 I/Os PER TRANS.	DESIGN #1 TOTAL I/O	DESIGN #2 I/Os PER TRANS.	DESIGN #2 TOTAL I/O
SALES-POSTING	200.0	57.0	11,400.0	80.5	16,100.0
SALES-REPORT	0.8	7400.0	5,920.0	4400.0	3,520.0
			17,320.0		19,620.0

The table above shows that design #1, the one without the search path on PURCHASE-DATE will perform less total I/Os than design #2. The example we have just gone over, although overly simplified illustrates one type of calculation that a simple design evaluation tool can perform. Such a design evaluation tool would also calculate storage requirements for two or more alternative designs.

More sophisticated analytical modelling tools, based on queueing theory [Sevcik] are now starting to be developed to predict data base performance. Such tools would be used to more accurately predict data base performance taking into account the effects of different transactions running concurrently on the system, interaction with other activity on the system, data base locking strategies, disc caching, etc.

2.6.2 Data Base Prototyping Tools

Another approach that can be used to evaluate a data base design is to build prototypes of the most active transactions and actually measure the performance of these transactions using performance measurement tools. There are several tools that can assist with the different tasks involved in this approach to data base design evaluation. Such tools are not only useful in the initial data base design process, they are also useful in the ongoing data base performance-tuning and maintenance activities. These tools include :

- **Application prototyping tools.** Currently, there are tools available on the HP3000 such as the TRANSACT language, a part of the RAPID/3000 package, and some third party software packages that allow you to quickly build prototype applications. An effort needs to be made to tie together the transaction models used by the design tools to these prototyping tools.

- **Data base simulation tools.** An example of such a tool is IDEA (IMAGE Data base Evaluative Analyzer), a contributed library software package that simulates data base activity, and generates timing reports. Data base activity is specified through scripts which describe transactions against the data base. IDEA actually generates calls to the IMAGE intrinsics to obtain its performance estimates. It would be useful if scripts for an IDEA-like tool could be automatically generated from the transaction descriptions used as input to the design tools.

- **Performance monitoring tools.** Tools are needed to monitor data base activity. Such tools should monitor and generate reports detailing how often each data set was accessed, what type of access was performed, what search path was followed, how much contention for data base resources was experienced (e.g. how often was a data set being accessed locked by another transaction), and so on.

- **Test data generation tools.** A test data generator should automatically load a data base with random data which satisfies characteristics of the real data as specified by the designer. For example, a designer should be able to specify that a detail data set search item field be loaded with randomly distributed data in the range of 100,000 to 200,000 with an average chain length of five.

- **Data conversion tools.** These tools would simplify the task of loading a new data base with existing data, perhaps contained in a KSAM file, or in another IMAGE data base.

- **Data base restructuring tools.** An example is the ADAGER package which allows IMAGE data bases to be restructured online, without having to unload the data to do the restructuring.

3. DESIGN TOOLSET

In this section I would like to briefly discuss how a design toolset might integrate the different tools described in this paper. Such a toolset would provide a common interface to the different tools, would provide some common functionality across the different tools, and would allow the different tools to communicate with each other.

The central component of such a design toolset is going to be a data dictionary. The data dictionary is the interface through which the different tools communicate. The data dictionary is going to have to maintain different representations (E-R models, transaction models, network models, and relational models) of the meta-data describing the different models that are used by the design tools in the different phases of the design process.

The design toolset should provide (through the data dictionary) some common functionality supporting the design process. This functionality would include :

- **Documentation.** All the tools should allow all design decisions (from requirement specification through physical design) to be thoroughly documented.
- **Logging.** Changes to a design should be logged to permit review of the steps taken to arrive at a current design.
- **Version Management.** Make it easier for designers to explore alternative designs.
- **Archiving.** Ability to store material (e.g. local E-R models) generated during the design process which is not needed in a production dictionary on backup storage.

It is also important that each tool be designed to perform as specific and small a task as possible. This will allow tools to be used either independentlv or in combinations without interactions. For example, it should be possible to use the design evaluation tools without having to first use the modelling, schema mapping, and physical design tools. This allows the design evaluation tools to be used to evaluate the impact of design changes to an existing data base.

4. REFERENCES

Batini — Batini, C., Lenzerini, M., and Santucci, G., "A Computer-Aided Methodology for Conceptual Data Base Design". Information Systems, Vol. 7, No. 3, 1982, pp. 265-280.

Ceri — Ceri, S., Navathe S., and Wiederhold, G., "Distribution Design of Logical Database Schemas". IEEE Transactions on Software Engineering, Vol. SE-9, No. 4, July 1983, pp. 487-504.

Chen — Chen, P.P., "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36.

Chen2 — Chen, P.P., and Yao, S.B., "Design and Performance Tools for Data Base Systems". Proceedings 3rd International Conference on Very Large Data Bases, 1977, pp. 3-15.

Hubbard — Hubbard, G.U., *Computer-Assisted Data Base Design*, Van Nostrand Reinhold Co., 1981

IEEE — IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, January 1977.

Kahn — Kahn, B.K., "A Method for Describing Information Required by the Data Base Design Process". Proceedings ACM-SIGMOD International Conference on Management of Data, 1976, pp. 53-64.

Kent — Kent, W., "A Simple Guide to Five Normal in Relational Database Theory". Communications of the ACM, Vol. 26, No. 2, February 1983, pp. 120-125.

Sevcik — Sevcik, K.C., "Data Base System Performance Prediction Using an Analytical Model". Proceedings 7th International Conference on Very Large Data Bases, 1981, pp. 182-198.

Teorey — Teorey, T.J. and Fry, J.P., "The Logical Record Access Approach to Database Design". ACM Computing Surveys, Vol. 12, No. 2, June 1980, pp. 179-211.

Yao — Yao, S.B., Navathe, S.B., and Weldon, J-L, "An Integrated Approach to Database Design". Proceedings 1978 NYU Symposium on Data Base Design (Lecture Series in Computer Science), no. 132, pp. 1-30, Springer-Verlag, 1982.

Abe Lederman was born in Montevideo, Uruguay and immigrated to the U.S. in 1968 at the age of 10. He received the B.S. and M.S. degrees in computer science from the Massachusetts Institute of Technology in 1981.

Upon graduation, Abe Lederman accepted a position as a member of technical staff in Hewlett-Packard's Information Network Division working on maintaining and enhancing the RAPID software package. He is now working in the data base section on data base tools.

RPG/3000

by Nancy Lucas
Marketing Engineer
Hewlett-Packard

Accessing and controlling a file that is open only to you is a relatively simple matter. However, when the file is being accessed by several users simultaneously, each user must be aware of how access is controlled for this shared file.

Simultaneous Access of Files

When a program issues a request to open a file, that request is regarded as an individual accessor of the file and a unique file pointer, set of buffers, and other file control information is established for that access path. Even when the same program issues several different open calls for the same file, each call is treated as a separate accessor. Under the normal security provisions of MPE, when an accessor opens a file not presently in use, the access restrictions that apply to this file for subsequent accessors depend upon the access mode requested by this initial accessor.

File sharing restrictions may be specified using file equations.

The file sharing restriction options are:

;EXC	Exclusive Access	After file is opened, PROHIBITS concurrent access in ANY mode through another open request, whether issued by this or another program until this program issues a close or terminates.
;SEMI	Semi-Exclusive Access	After file is opened, PROHIBITS concurrent write access through another open request, whether issued by this or another program, until this program issues a close or terminates. PERMITS concurrent read access.
;SHR	Sharable Access	After file is opened, PERMITS concurrent access to file in any mode through another open request issued by this or another program, in this or any other session or job.

Exclusive Access

```
:FILE A;EXC
```

In all cases, when the first accessor to a file opens it with Exclusive Access, all other attempts to open the file will fail.

This option is useful when you wish to update a file, and wish to prevent other users or programs from reading or writing to the file while you are using it. Thus, no other users can read information that is about to be changed, nor can they alter that information.

Semi-Exclusive Access

```
:FILE A;SEMI
```

This option allows other accessors to read the file but PREVENTS them from altering it.

Share Access

```
:FILE A;SHR
```

This option allows other accessors to use the file. Each accessor transfers its input/output to and from the file via its own unique buffers, using its own set of file control information and its own record pointer. Effectively, each accessor retrieves its own copy of that portion of the file presently in its buffer.

File sharing by two or more processes may be hazardous. When a file is being shared by two or more processes and is being written to by one or more of them, care must be taken to ensure that the processes are properly interlocked. The necessary interlocking is provided by properly locking and unlocking the file.

Locking and unlocking in the MPE system allows you to perform your own conditional or unconditional locking and unlocking on Image, KSAM, and MPE files. When an unconditional lock is executed on a file which cannot be locked immediately, the calling program suspends until the file can be locked. A conditional lock will take place only if the file is not currently locked. If the file is locked, control returns immediately to the calling program, and the lock fails.

One example of unconditional locking occurs when a file is shared between a writing process and a reading process, with the writing process adding records to the file. Locking is executed prior to writing each record. Unlocking is then executed when the writing process is finished. By contrast, the reading process locks the file prior to reading each record, and unlocks the file after reading is finished. If the writing process should execute while the reader is in the middle of a read, the writer's call to lock the file will be suspended until the reader signals that it is finished by unlocking the file.

Locking within RPG

In order for locking and unlocking to be allowed for a shared file in RPG you must enable the appropriate locking facility. For KSAM and MPE files, you enable the MPE dynamic locking facility by specifying a KLOCK or KNOLOCK continuation record for the file. For Image files, you enable Image locking by specifying one of the locking modes (B,S,1,9,R, or L) on the KIMAGE continuation record for the file.

KSAM and MPE

You can only lock KSAM and MPE files on the file level (i.e., the entire file must be locked; indivdual records cannot be locked). Both automatic and manual locking options are available.

Automatic Locking

When the KLOCK option is specified, RPG opens the file with the dynamic locking facility enabled for shared access and automatically locks and unlocks the file whenever a record is read or written.

Manual Locking

When the KNOLOCK option is specified, RPG opens the file with the dynamic locking facility enabled, but does not do any automatic locking or unlocking of the file. If you want to lock and unlock the file, you will need to do so manually by using the LOCK and UNLCK operations in the Calculation Specifications.

If one user opens a file with the dynamic locking facility enabled then all other concurrent users must also open the file with dynamic locking enabled, whether or not they are going to be locking and unlocking the file.

There are 3 ways that concurrent users can enable the dynamic locking facility:

1) Through use of a KLOCK continuation record in RPG program.

2) Through use of a KNOLOCK continuation record in RPG program.

3) Through use of the LOCK option on a file equation.

RPG does automatic locking and unlocking of KSAM, MPE, and Image files in the following way:

An input file is locked before it is read and unlocked after it is read.

An output file is locked before it is written and unlocked after it is written.

An update file is locked before a record is read, and unlocked either after it is updated or before the next lock and read. That is, if an update file has been locked and read, but not updated, RPG unlocks the file when the program next attempts to lock and read from it.

```
INPUT FILE:  LOCK-->READ------------->UNLOCK->-------->LOCK...
OUTPUT FILE: LOCK----------->WRITE--->UNLOCK->-------->LOCK...
UPDATE FILE: LOCK-->READ---->UPDATE-->UNLOCK->-------->LOCK...
      or     LOCK-->READ--------------------->-UNLOCK->LOCK...
```

igure 1

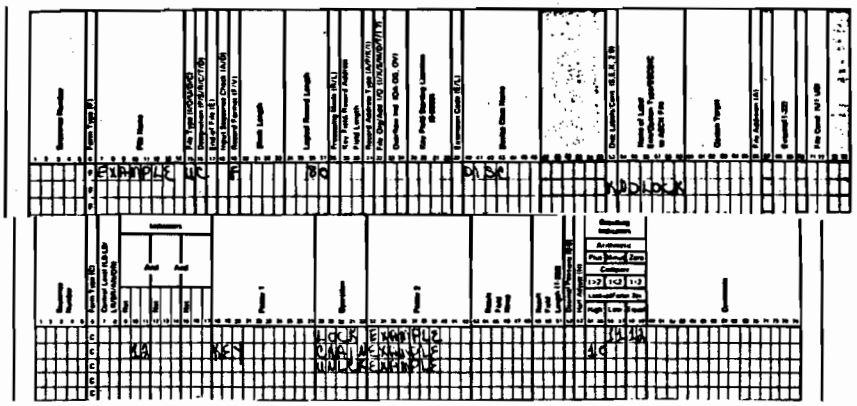
NOTE - If a user uses the LOCK option on a file equation, and does not also include a KLOCK or KNOLOCK continuation record, RPG will not perform automatic locking and will not recognize any LOCK and UNLCK operations found in the Calculation Specifications. Therefore, this combination (LOCK on file equation, without KLOCK or KNOLOCK in program) serves only to open the file with the dynamic locking facility enabled to allow concurrent access to a shared file.

The following figure shows how multiple programs can access a file concurrently.

Program 1 is doing updates.
Program 2 is writing.
Program 3 is reading.

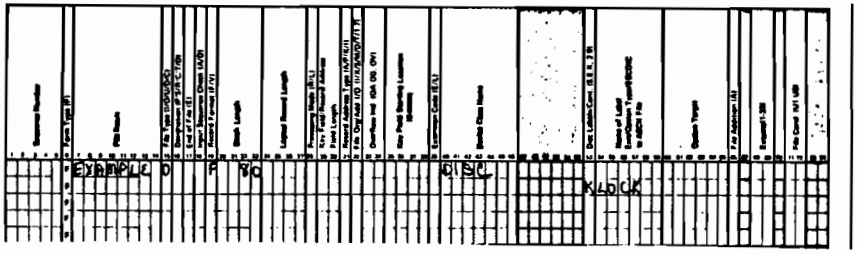
PROGRAM 1

:FILE EXAMPLE;SHR



PROGRAM 2

:FILE EXAMPLE;SHR



PROGRAM 3

:FILE EXAMPLE;LOCK

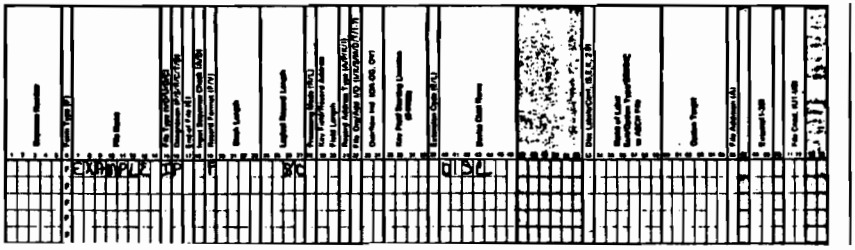


Figure 2

LOCK/UNLCK Summary for KSAM and MPE files

Type of Lock	Factor 1	Operation	Factor 2	Result Fld
KSAM file	blank	LOCK/UNLCK	filename	blank
MPE file	blank	LOCK/UNLCK	filename	blank

Figure 3

KDSNAME continuation records are used to link several access paths to a single physical file. Only the first File Description of such a DSNAME group will be used to determine whether or not the file is to be opened with dynamic locking enabled. To enable dynamic locking for a DSNAME file, then, you must specify a KLOCK or KNOLOCK continuation record for the first File Description of the DSNAME group, or include a ;LOCK option on the file equation for the first file in the group.

IMAGE

To lock Image files you specify one of the locking modes (B,S,1, 9,R, or L) on the KIMAGE continuation record. Image files can be locked at the data base, data set, or data record (item) level. You have the option of doing automatic or manual locking.

Automatic Locking

MODE B - The data base is locked for the duration of program execution.

MODE S - A specified data set is locked for the duration of program execution.

MODE 1 - The data base is locked and unlocked whenever a record is accessed from that data base.

MODE 9 - A data set is locked and unlocked whenever a record is accessed from that data set.

MODE R - A specified record is locked and unlocked whenever it is accessed.

NOTE - All of the above locking modes do UNCONDITIONAL AUTOMATIC locking.

Locking modes B and S cause a data base or data set to be locked at program initialization and remain locked for the duration of program execution. Modes 1,9, and R cause the data base, data set, or record to be locked and/or unlocked whenever a record is read, written or updated.

See FIGURE NLTXT-1 to determine how RPG automatically locks and unlocks files.

Manual Locking

MODE L - Allows the user to do manual locking and unlocking (conditional or unconditional) via the LOCK and UNLCK operations in the Calculation Specifications.

To manually lock and unlock a data base, data set, or data record you specify LOCK or UNLCK in the Operation Field, the filename in Factor 2, and Resulting Indicators. For data base locking, the data base name goes in the Result Field, and a value between 1 and 256 must be specified in the Result Field Length (1

is recommended). The entry in the Result Field Length is neccessary because the Result Field is interpreted by RPG as a field rather than a literal enclosed in quotes. Therefore, in addition to providing the data base name for the LOCK and UNLCK operator, the Result Field Name and Length define an actual RPG field. This field will appear on the compiler listing Symbol Map and Cross Reference. For data record locking a key value goes into Factor 1.

LOCK/UNLCK Summary for Image files

Type of Lock	Factor 1	Operation	Factor 2	Result Fld	Result Fld. Lth
IMAGE Data Base	blank	LOCK/UNLCK	filename	data base	1 - 256
IMAGE Data Set	blank	LOCK/UNLCK	filename	blank	blank
IMAGE Record	key value	LOCK/UNLCK	filename	blank	blank

Figure 4

If you are manually locking and unlocking KSAM, MPE, or Image files, Resulting Indicators which both define the type of locking to be done (conditional vs. unconditional) and return status information for the operation must be specified. The High indicator is optional, but one of either the Low or Equal indicators is required. The presence of the High indicator declares conditional locking whereas, its absence declares unconditional locking.

Conditional and Unconditional Locking on Image Files

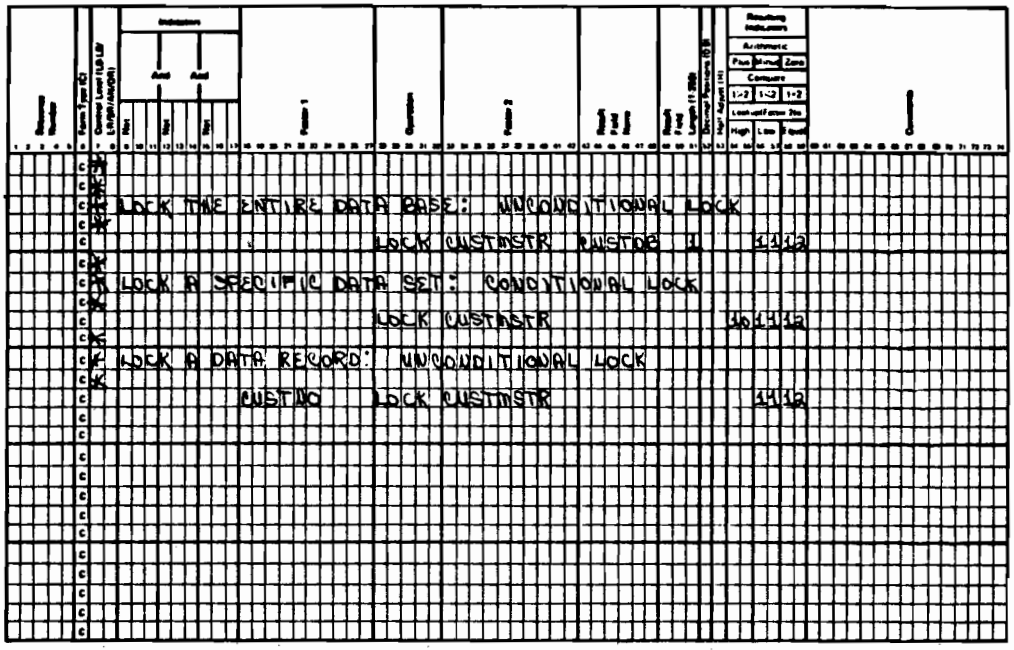


Figure 5

RPG Enhancements

Some of the recent enhancements to RPG include Forms Downloading on 2624B terminals, and Buffer Integrity Checking. Forms Downloading is enabled via a KFORMDL continuation record. You are able to download 1 to 255 forms. Buffer checking is an enhancement that allows you to ensure data integrity for KSAM and MPE files. Other enhancements to RPG are being done to make RPG/3000 more Sys/34 compatible.

Nancy Lucas is a Marketing Engineer in the Computer Language Lab at Hewlett Packard. Born and raised in Virginia, she received her B.S. degree in Computer Science at Old Dominion University. Upon completion of her degree, Nancy relocated to California where she has been employed at Hewlett Packard for 2 1/2 years. She is currently supporting APL, RPG, and BASIC on the HP 3000.

LOCK/UNLCK Resulting Indications

Resulting Indicator Set ON	IMAGE				EDAM and MPE	
	LOCK			UNLCK	LOCK	UNLCK
	Data Base	Data Set	Record			
High (Conditional Locking only)	Status=20 — Data base locked or contains locks	Status= 20—Data base locked or contains locks 22—Data set locked by another process 23—Entries locked within set	Status= 20—Data base locked or contains locks 22—Data set locked by another process 23—Entries locked within set 24—Item conflicts with current locks 25—Entries already locked	Status > 0 — Exceptional error	Condition Code > —Locked by another process	Condition Code > —Not already locked
Low	Status < 0 — File System or memory manager failure	Status=-186 — Second lock without CAP=MR	Status=-186 — Second lock without CAP=MR	Status < 0 — File system or memory manager failure	Condition Code < —Not opened with dynamic locking facility enabled or need MR capability	Condition Code < —Not opened with dynamic locking facility enabled or need MR capability
Equal	Status=0 — Request granted	Status=0 — Request granted	Status=0 — Request granted	Status=0 — Request granted	Condition Code= — Request granted	Condition Code= — Request granted
None	Status=any value other than above	Status=any value other than above	Status=any value other than above	Can never happen — will always have one Resulting Indicator ON	Can never happen — will always have one Resulting Indicator ON	Can never happen — will always have one Resulting Indicator ON

Figure 6

Avoiding
Heap Overflow and Stack Overflow
with Pascal/3000

by Christopher P. Maitz
Hewlett-Packard

This paper is an attempt to share with the Pascal/3000 community things that were learned in the development of the Pascal/3000 compiler. The compiler is written in Pascal and is compiled with itself. The solutions we found to avoid heap overflow and stack overflow while using the Pascal/3000 compiler should be helpful to other developers of Pascal systems.

The Pascal/3000 project team has been facing the problem of lack of space during compilation since the early days of the Pascal/3000 compiler's development. We currently use the Pascal/3000 compiler to compile its own source, but before we reached the stage in development when enough features existed in Pascal/3000 to make self-compilation possible, we compiled our source with the "P4" compiler available in the contributed library. The P4 compiler took Pascal source code and produced P-code. An assembler then converted the P-code into SPL/3000 source code, at which point the SPL/3000 compiler was used to produce 3000 code. This painful process was a great motivating factor for us to develop the compiler to the point at which we could compile the compiler with the compiler. As a foreshadowing of future woes, we faced the space problem even with the P4 compiler. My first assignment as part of the Pascal/3000 team was, in fact, to enhance our version of the P4 compiler so that its symbol table would pack two characters per word rather than one, doubling the size of its symbol table.

What exactly is the stack size limitation problem? A program executing on the HP3000 uses a stack as shown in Fig. 1. Primary and Secondary DB contain the global variables. When a procedure is called, its activation record (stack marker and local variables) is pushed onto the stack. The set of stack markers between QI (Q Initial, the stack marker for the outer block) and Q (the stack marker for the currently executing procedure) shows at any given time the dynamic calling sequence of the program. The area between Q and S (the top of stack) represents space allocated for the current procedure's local variables and any expressions undergoing evaluation. Between S and Z (the stack limit) is space for the expansion of S.

The other side of the stack is the area between DL and DB. This area can be thought of as free space as far as the system is concerned. For Pascal/3000 programs, however, this area contains the heap, where storage is allocated dynamically. Whereas the system automatically increases Z as space is needed above S, the Pascal runtime library heap routines control the size of the DL-DB area directly, using the intrinsic DLSIZE. The total size of the regions both above and below DB must not exceed the stack segment size, which is determined when the system is configured. The maximum size is 31232 words. (Actual space available to the program is less, due to process information that the system keeps in the PCBX in the user's stack.)

When the Pascal/3000 compiler requests data to be allocated in the heap and not enough space exists in the stack, the Pascal runtime library traps the condition and the error

SYSTEM RESOURCE EXHAUSTED 1, COMPILE TERMINATED (426)

is generated and the compilation is terminated. When the Pascal/3000 compiler runs out of space at the other end of the stack (due to a procedure call, allocation of local variables, expression evaluation), an error also occurs. This error, unfortunately, cannot be trapped as the heap overflow error can. Instead of a Pascal error message, the system aborts the compiler and generates the message

```
ABORT :PASCAL.PUB.SYS.<segment-#>.<segment-offset>
PROGRAM ERROR #20 :STACK OVERFLOW
```

Both of these overflow situations will be referred to simply as the "stack size limitation problem" or "overflow problem" in the rest of the paper.

It is the responsibility of the runtime library to balance the areas on either side of DB. When space is needed in the heap and not enough space is left between the internal Top-of-Heap pointer and DL, DLSIZE is called to increase the area between DL and DB. When space is deallocated through use of the Pascal pre-defined heap routine RELEASE, DLSIZE is called to cut back this area, allowing the system more room with which to increase Z. The stack segment of a Pascal/3000 program is then split between the space used on either side of DB. The compiler has very few static data structures; its structures are almost entirely dynamic, the sizes of which depending significantly on the structure and character of the user's program. The Pascal/3000 compiler as it runs will generally require only a few thousand words between DB and Z, leaving the rest for the heap.

The compiler uses the dynamic structure of the heap to its advantage. When compiling a program the compiler processes the global information first, which includes labels, constants, types, variables and external procedure headers. Even though space is allocated for these dynamically in the heap, it remains allocated throughout the entire compilation since global identifiers can be referenced anywhere in the program.

Processing procedures is very different from processing the outerblock and therein lies the effectiveness behind several key space-saving ideas that will be presented shortly. A procedure's identifier and parameter list are stored in the heap, along with other global information. Before the declarations and the body of the procedure are processed, a MARK is done in the heap which saves the Top-of-Heap pointer (Fig. 2.a). The procedure's declarations and statements are then processed and stored in the heap (Fig. 2.b), but, after the code for the procedure is generated, all of the space allocated since the MARK can be reclaimed using RELEASE (Fig. 2.c). The heap is then returned to its state before the declarations and statements of the procedure. All that remains is the procedure's name and descriptions of its parameters, which is only the information necessary to call that procedure from other points in the program.

With this short introduction to the workings of the compiler, one can see that two ways to make the 3000 stack stretch further would be to reduce the amount of global data (the space which remains during the entire compilation) and to keep down the size of individual procedures. Several approaches to the global problem will be presented first.

1. GLOBAL/EXTERNAL Compiler Options. In a normal program (neither $GLOBAL$ nor $EXTERNAL$ specified) the compiler assumes that all global variables are present and therefore assigns addresses to them in Primary DB. For all other compilation units which comprise the program, the same assumption is made. Obviously, if the compiler is assigning addresses for the global variables, they must all be declared in each compilation unit and in the same order to ensure that the addresses used are the same for each compilation.

This requirement is relaxed with the use of the $GLOBAL$ and $EXTERNAL$ compiler options. As before, the compilation unit which contains the outer block must have all of the program's global variables declared within it. The compiler option $GLOBAL$ is used for this compilation unit to tell the Pascal/3000 compiler that 1) all of the program's global variables are present, 2) the compiler should assign addresses to the global variables and 3) special information should be put into the USL file regarding these variables. This information consists of the name and address of every global variable. In the other compilation units of the program, marked with the $EXTERNAL$ option, the compiler does not assign addresses to the global variables. When it generates code which references a global variable, it doesn't use an address in the instruction; instead, it leaves the address part of the instruction blank and puts that instruction on a list with any other instructions which accessed that variable. For each global variable, the compiler puts into the USL file a special entry with the variable's name and a link to the list of its references. When the USL file is prepared into a program file, the MPE Segmenter gets the actual address of the variable from the information generated by the outer block compilation and uses it to fix up all of the instructions which actually reference that variable in the other compilation units. In this way, the binding between a variable's name (the ways it is referenced at the user level) and

its address (the way it is referenced by the computer) is postponed until PREP time, allowing the compiler to relax the need to see all of the global variables in all of the compilation units.

Not only does this approach yield significant savings, it can also be implemented as needed on individual compilation units. All of the compilation units that comprise the Pascal/3000 compiler use $EXTERNAL$, but only those that have experienced heap or stack overflow problems include subsets of the global variables. Compilation units which have never had problems are still compiled using all of the global variables. This method was only used on the units that needed it as they needed it.

To implement this approach, each compilation unit should have a separate file for the global variables that it uses. Just include the global types and a file containing only the global variables used in the compilation unit (Fig. 3). This method requires that global variable names should be unique to 15 characters, a limit imposed by the MPE Segmenter.

```
$EXTERNAL$
PROGRAM CmpUnit1(Input,Output);

   {The global constants and types}
$INCLUDE 'GlobTyps.Types.Account'$

   {Just the global variables that this compilation unit uses}
$INCLUDE 'CmpUnit1.Vars.Account'$

   {Include the procedures for this compilation unit here}
.
.
.

BEGIN   {Empty outer block}
END.    {Empty outer block}

*** Now the outer block compile ***

$GLOBAL$
PROGRAM MainProg;

   {The global constants and types}
$INCLUDE 'GlobTyps.Types.Account'$

   {All of the global variables}
$INCLUDE 'GlobVars.Vars.Account'$

BEGIN  {Actual outer block}

   {Code for the outer block}

END.   {Actual outer block}
```

Fig. 3. Use of $GLOBAL$ and $EXTERNAL$ Compiler Options

2. External Procedure Declarations. Perhaps the greatest savings in stack space can be achieved by declaring external procedures only on the level used. Although the most natural place to declare external procedures may be the outer block, there is no requirement that they be so declared. (Even though not declared in the outer block, the compiler knows that they are external to the program.) External procedures may be declared on any level and the suggestion here is to declare them only at the level used. The source code for the Pascal/3000 compiler consists of about fifty separately compiled programs. During one stage of development, each of these compilation units had from 20 to 100 external procedures declared globally. An individual procedure rarely referenced more than five of these external procedures, yet space used to describe the name and parameters for all of procedures was

globally allocated since they were declared in the outer block. By moving these external procedure declarations inside the scope in which they were referenced, the Pascal/3000 team was able to reclaim a significant amount of space.

An example serves to illustrate the potential savings. Assume the following for a particular compilation unit:

- 100 external declarations

- 3 parameters per external declaration (average)

- 5 external procedures referenced by each level one procedure in the compilation unit (average)

- each procedure identifier requires 14 words of space

- each parameter identifier requires 8 words of space

The sizes for procedure and parameter identifiers are given in HP3000 16-bit words. The parameter specification in an average procedure declaration would then be

```
      3        *        8          =      24

   # of          space for          total
parameters     a parameter       space used
                                for parameters
```

Therefore, the amount of space consumed by a single external declaration is

```
      14       +      24          =      38

  space for      space for          total
procedure id.   parameters        space used
```

If all of the external procedure declarations are declared in the outer block, the total space consumed is

```
     100       *      38          =     3800

    # of          space for          total
externals      external decl.     space used
```

Now if externals are only declared at the level in which they are referenced, space consumption is

```
      5        *      38          =      190

    # of          space for          total
externals      external decl.     space used
per procedure
```

In this example, over 3600 words of stack space were saved by declaring the external procedures locally.

The value of this approach depends on the structure of one's program. If there exists a procedure in the compilation unit which references all of the externals used in the entire compilation unit (or if the entire compilation unit is a single procedure), this method will be of no benefit. Modularized programs with procedures oriented toward single functions will realize the most gain by declaring external procedures at the level in which they are used.

Another advantage of this approach towards saving space is that it can be applied to individual compilation units as needed. Removing all external declarations from the outer blocks of all the compilation units for the Pascal/3000 compiler would have been a large effort. Instead we only used this approach for those compilation units which were experiencing stack overflows during compilation.

Rather than imbed in-line declarations of external procedures into your code, it is

suggested that you create one file for each external procedure header and place these files in a special group. (We called this group "EXT".) If you need to use an external procedure, then include the file which has the external procedure declaration in the procedure which calls it (Fig. 4). The viability of putting each external into its own file depends on your naming conventions. It was a natural manner in which to organize external declarations for the Pascal/3000 project, since each procedure in the compiler already resided in its own file in a special group for procedures (called "PROCS"). However it is done, localizing external procedure declarations can yield significant savings in stack space.

```
PROCEDURE Proc1 (Parm: Integer);

   VAR
      I : Integer;

   $INCLUDE 'InsertNu.Ext.Account'$
   $INCLUDE 'DeleteNu.Ext.Account'$

BEGIN  {Proc1}
   .
   .
   .
InsertNumber (I);   {an external procedure call}
   .
   .
   .
DeleteNumber (I);   {an external procedure call}
   .
   .
   .
END;   {Proc1}
```

Fig. 4. Local Declaration of External Procedures

3. Odds and Ends. Here a few general tips which the Pascal/3000 team learned during development. If your procedure or function spans more than 2 pages (without comments) you may try to break part of it up into one or more nested procedures. Procedures that are too long tend to be hard to read, are difficult to debug, take a lot of stack space to compile, and generally fail to follow structured programming guidelines. If a case element statement is long, make it a nested procedure to save space. If there is a large number of cases in the case statement, break it up into smaller ones. Put the more frequent case elements in the first part of the case statement; in the OTHERWISE part of the case statement, have a procedure call to a nested procedure which has the rest of the case elements.

That brings to a close the methods actually used in the development of the Pascal/3000 compiler. There are, however, several more ideas that will be presented in brief.

4. If you are compiling with the option $TABLES ON$ and are at the end of the procedure and either the code offsets or code listing has been generated and the identifier table has not yet been generated, you most likely have a stack overflow in the routines that generate the tables. As a short term solution, you can insert the compiler option $TABLES OFF$ before the procedure to proceed with the compilation. If you really want an identifier map, try one of the other solutions.

5. Divide! Are you including a large number of constants that you don't use? For example, if you have one file for all error numbers, you may want to break them up into smaller files based on their functionality.

6. Divide! Try separating global types into files which reflect their functional independence. Include only the files needed to compile an individual compilation unit. You may also have to divide up the source into smaller units to take advantage of this. Also, if a source has many level-1 procedures, try dividing the source into separate compilation units, since then there would be fewer procedure headers (procedure names and parameter lists) declared.

7. Run the compiler specifying "NOCB" in the RUN command; this moves part of the system information stored in the PCBX to an area outside the program's stack.

```
:RUN PASCAL.PUB.SYS;PARM=7;NOCB
```

This will save you a small amount of space immediately, but it does not really solve the problem. Try one of the other solutions for more permanent results.

8. Use integer constants instead of enumerated types and then only include the specific constant values that are used in any particular procedure. To then declare a variable of that "type", it is only necessary to include the constants that denote the starting and ending values of the range (Fig. 5).

```
PROGRAM Prog;

CONST
   FirstError = 0;
   LastError = 100;
TYPE
   ErrorNumber = FirstError..LastError;
.
.
.
PROCEDURE Proc1(Pointer : PtrType);
   CONST
      $INCLUDE 'ErrFile1.Const.Account'$
   VAR
       Error : ErrorNumber;
   BEGIN
   IF Pointer = NIL
   THEN
      Error := BadPointer;  {BadPointer is in ErrFile1}
   END;

BEGIN   {outer block}
.
.
.
END.    {outer block}
```

Fig. 5. Replacing Enumerated Types with Constants

This solution is recommended only as a last resort, as it asks the programmer to sacrifice the use of enumerated types, a very nice feature of Pascal. One of the other approaches should be tried first, since most of them maintain or improve the program's style and organization.

9. If one procedure is too big to compile and there is a good reason for not breaking it up into smaller procedures, you may try moving the procedure so that it is compiled earlier in the compilation unit. This will help a little since the declarations of the other procedures in the unit will be processed after the offending procedure.

Recommendations

Not all solutions cited here are appropriate for every project and every circumstance and stage of development. Below is a quick guide for finding a method to suit your particular situation.

If you are just starting development of a large system, consider using the following solutions from the outset: 1, 2 and 3.

If you are having a problem with just one procedure and want a quick solution to get it compiled, try one or more of the following: 3, 4, 7 or 9.

If you are interested in saving space on all of your compiles and are willing to spend some time restructuring your programs, try one or more of the following: 1, 2, 5, 6 or 8.

Summary

This paper has presented a number of ways that the stack size limitation can be avoided when using the Pascal/3000 compiler. It is the hope of the author that the approaches outlined here may help the Pascal programmer to more easily develop large systems on the HP3000.

Acknowledgments

I would like to thank Ron Smith for his help in compiling the data for this paper, Sue Kimura for her daily encouragement while it was being written and the Pascal/3000 project team, both past and present, for the opportunity to work and learn with them on the project.

References

MPE IV System Manager/System Supervisor Reference Manual, HP3000 Computer Systems, Hewlett-Packard Company, 1981, Part. No. 30000-90014.

Pascal/3000 Reference Manual, HP3000 Computer Systems, Hewlett-Packard Company, 1981, Part No. 32106-90001.

System Reference Manual, HP3000 Computer Systems, Hewlett-Packard, 1978, Part. No. 30000-90020.

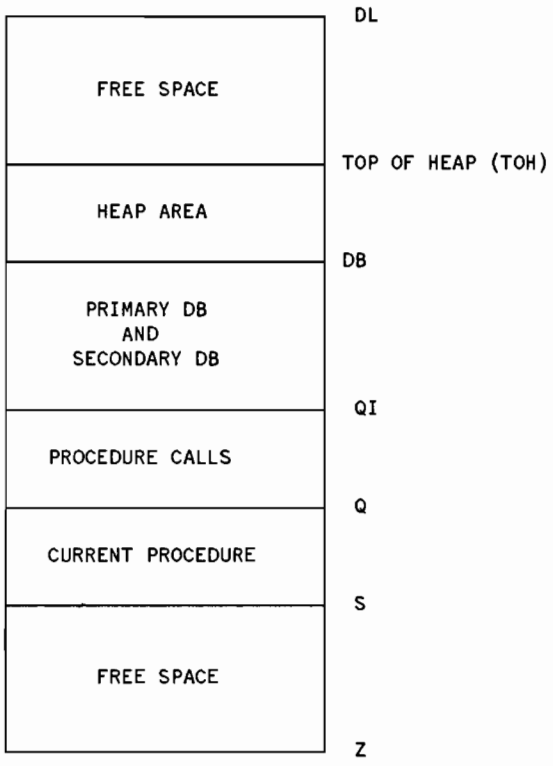
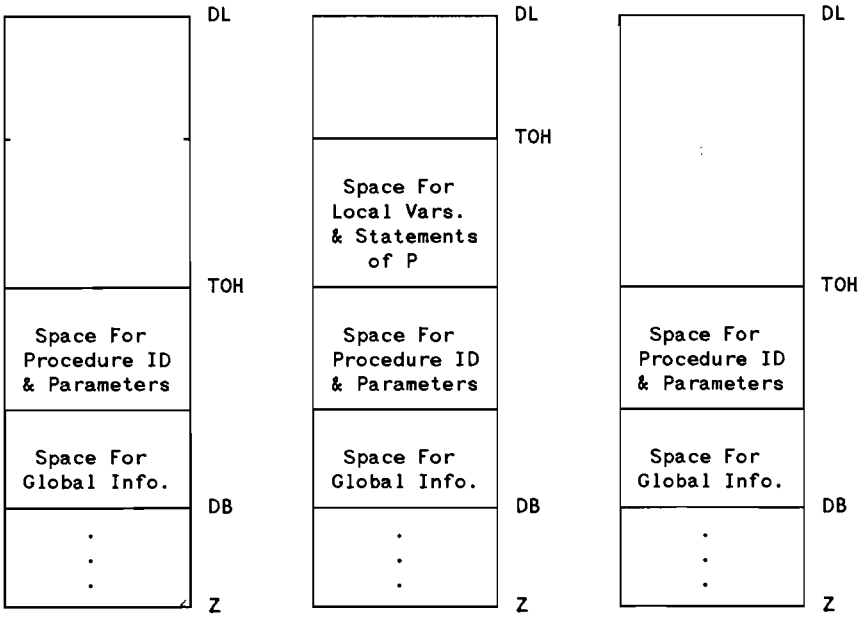


Fig. 1. Pascal/3000 Compiler Runtime Stack



a. MARK saves Top_Of_Heap

b. After P processed

c. RELEASE after code generated

```
            PROGRAM Outerblock;

            VAR
               X, Y : Integer;

            PROCEDURE P ( Parm1, Parm2, Parm3 : Integer);
a. →           VAR
                  Local1, Local2 : Integer;
               BEGIN  {Body of P}
               .
               .
               .
b. →           END;   {Body of P}

c. →        PROCEDURE Q;
               .
               .
               .
```

Fig. 2. Heap Activity of Pascal/3000 During Compilation

Biographical Sketch

Name : Christopher P. Maitz

Title : Member of Technical Staff

Employer : Hewlett-Packard Computer Languages Lab

Job Responsibilities : Involved in the development and enhancement of the Pascal/3000 compiler since 1979.

Education: BS, Mathematics, Carnegie-Mellon University

Marital Status : Married

Satellite Communication: Overview and Application

by Doug McLean

Commercial use of satellite links(or space segments) is a rather old technology by computer industry standards. TELSTAR I, the first commercial satellite was launched in 1962. INTELLSAT has been providing international satellite communication services since 1965. Recently the cost of earth stations and transponder time has dropped, making satellite communication feasible for HP 3000 users. The purpose of this paper is to look at the state-of-the-art of satellite data communication and to review an example of how it has been used at HP to lower data communication costs.

I. Satellite Communication Terminology

Before proceeding with a discussion of the satellite communication industry, it will be helpful to review some of the "buzz words" typically used in the industry.

The Hardware

A satellite link or "space segment" is made up of three parts: the satellite(or bird), the antenna(or dish), and the attendant signal processing equipment which transforms the data from the computer into a form the satellite can handle. The dish and attendant signal processing equipment collectively make up the earth station.

To the Stars and Back

When the earth station broadcasts to a satellite, it does so on a circuit known as the up-link at a frequency of approximately 6 GigaHertz(GHz). When the signal reaches the satellite, it is processed by a piece of hardware known as a transponder. The transponder does three things to the signal. First, it amplifies it so it can be received more easily(and accurately) by the receiving earth station. Second, it changes the frequency of the signal to approximately 4 GHz. Finally, the transponder broadcasts the signal back to the receiving earth station on the down-link. Because of the frequency shift performed by the transponder, a single earth station can transmit and receive simultaneously at very high data rates.

Satellites

There are currently 27 communication satellites in orbit serving the northern half of the western hemisphere. In addition there are fourteen INTELSAT satellites doing international communication world wide. All of these satellites are 22,300 miles from the equator in geostationary orbit. This means that their position relative to the earth never changes. Because it is in geostationary orbit a satellite is available 24 hours per day, 365 days per year with the exception of outages due to solar eclipse. These outages are generally quite brief(averaging 7 minutes) and occur once per day for a week in the spring and fall. The outages do not generally present any operating problems as they are extremely predictable. One of the more heavily used domestic communication satellites is Western Union's WESTAR IV. WESTAR IV has twelve transponders on board of which two are used as back-up in the event one of the other ten fails. Each transponder has a total bandwidth capacity of 36 MHz, which means that each transponder may be used to carry:

-One color television broadcast
or
-1200 voice channels
or
-A data channel of 50 Mbps

Clearly, with this much bandwidth, very few users need to own a transponder. Consequently, a number of companies have purchased one or more transponders on WESTAR IV and resell the transponder time on an occasional use basis. These companies include Vitalink, AMSAT and Equatorial Communication.

Propagation Delay

Because communication satellites are stationed 22,300 miles above the earth, it takes 135 milliseconds for a signal to travel from the ground to the satellite. Thus, when one bit of data leaves a computer in San Francisco, it is at least 270 milliseconds later that it is received by an earth station in Boston. Therefore, a full round trip delay of approximately one-half second is normal in a satellite network. Fortunately, the earth stations are programmed to allow for this "propagation delay", so no data is lost. The delay can, however, be a nuisance in a highly interactive application. Good satellite applications tend to trans- mit large blocks of data to minimize some of the effects of propagation delay.

II. Satellite Industry Structure

Participants in the satellite industry fall into 4 broad categories:

Common Carriers: Those companies that lease transponder time. Signals are transmitted and received by dishes owned by the carrier and generally situated on the carrier's premises. While some common carriers have their own satellites, others have leased or purchased individual transponders from other common carriers. AMSAT, COMSAT, SBS, INTELLSAT and Western Union are all satellite common carriers.

Satellite and Antenna Construction: Electronics firms whose primary business is the design and fabrication of the hardware involved in satellite communication. Hughes Aircraft Company is currently the most popular satellite contractor and Scientific Atlanta and Harris Communications the most popular dish vendors.

Leased Private Network: Several firms such as RCA have purchased transponders from a common carrier and will lease "on-premises" equipment to users. The networking company also resells the transponder time it has available on a monthly or occasional basis. AMSAT, COMSAT AND RCA (under the brand name CYLIX) all offer private leased networks.

Owned Private Network: A few firms sell earth stations and network management services so a user can own his network. Only Vitalink offers the antenna, network management and the transponder time.

III. Network Design

As with any network, the topology chosen should be a function of the application. The purpose of this section is to review some of the more common topologies and how the currently available satellite technology can be used to implement them.

Public vs. Private

The first decision to be made in implementing a satellite network is whether or not to own the earth stations. For the very occasional user or the user whose nodes are all in major metropolitan areas, using the satellite common carrier's facilities can be quite economic. In many large cities (New York in particular) there are very few microwave rights of way left. As the microwave frequency spectrum overlaps the satellite C band(4/6 GHz) putting a private satellite network in such cities is often not possible. In these cases, it is generally easier and less expensive to interface the computers to the common carrier's equipment via high speed data line. Conversely, if a user needs to transfer large batches of data fairly frequently amongst sites which are rural or suburban, a private network can be extremely cost effective. A hybrid network which uses owned earth stations for receiving and the common carrier's facilities for transmitting should also be considered. Both of these alternatives are discussed below.

Mesh Topology

Perhaps the most flexible topology in

common use is the mesh topology. Mesh allows any node on the network to communicate directly with any other node. Each earth station sends data directly to the satellite. The satellite in turn broadcasts the data to all of the earth stations in the network. Each earth station then either processes the data or ignores, it depending on whether the data packet is addressed to it. A mesh topology is most appropriate for users with a need to move large volumes of data amongst a limited number of sites. Financially, a mesh satellite network become very attractive for sites more than 800 miles apart with a bandwidth requirement in excess of 56 Kbps. There are a number of ways for HP 3000 users to implement a mesh network. Companies like AMSAT lease on-premises dishes and profide end-to-end network management. Alternatively, users can implement a private satellite network by purchasing earth station equipment from one of a number of vendors (eg. Scientific-Atlanta). The problem with implementing a network this way is that a user then has to contract for transponder time with one of the common carriers. One earth station vendor, however, offers both earth stations and transpon- der time. Vitalink Communications Corporation purchased two transponders on WESTAR IV and resells the time to its customers on an occasional use or contract basis. Vitalink also provides network management services.

Broadcast Star Topology

For users who have a need to move large amounts of data from a central site to many remote sites and do not need response from the remote sites the broadcast star topology is more appropriate. Using this technique, data is collected and/or processed at a single site and then transmitted to a vendor's earth station. The data is then transmitted to the satellite and broadcast to the receiving earth stations. There are two variations on the broadcast star network. The most common method of implementation involves leasing a high speed terrestrial data link from the central data processing center to the vendor's earth station. Increasingly, however, users are purchasing their own transmitting earth station for transmission to the vendor's facility or directly to the receiving earth station. Depending on the type of broadcast protocols used, the receiving earth stations can be as small as two feet in diameter. The major problem with the broadcast star topology is that the transmitting station has no way of verifying whether or not the data has been accurately received. In applications where data accuracy must be verified an inexpensive, low speed terrestrial link back to the trans- mitting earth station can be used. RCA and Equatorial Communications both offer broadcast star networks for private use.

Economics of Satellite Communication

It is no great secret that earth stations are expensive. A small, low speed transmit and receive earth station costs approximately $50,000. A large, high speed station can cost upwards of a quarter million dollars. The payoff, of course, is that the relative operating costs are low compared to high speed terrestrial circuits. If a user needs to transmit large amounts of data, a satellite network, once installed, is extremely cost effective. Very few applications, however, require constant or even frequent use of a 56 Kbps data link. Justifying a satellite network often depends on calculating the aggregate data rate amongst the potential sites. A satellite network is capable of carrying data, voice, and video signals simultaneously. By multiplexing together a number of 9.6 Kbps or slower channels on a satellite network, a user can reduce her telephone, TYMNET and air travel bills. The next section contains a discussion of how Hewlett-Packard intends to do this internally. For those users who can not cost justify their own satellite network and do not want to use common carrier's facilities, the possibility of sharing earth station facilities may provide a viable alternative. By putting the earth station at a central location in a small to medium sized city and running high speed data links to the owners' facilities, a number of small users can realize the cost savings of an owned satellite network while sharing the start-up and operating expenses. Sharing earth station facilities is also becoming popular in large cities, as evidenced by New York's Teleport project. The New York City Port Authority in conjunction with Merrill-Lynch and Western Union have begun construction of a "dish farm" on Staten Island. The dishes will be connected to users in Manhatten via high speed fiber. As microwave rights of way are filled in other major cities we may see other Teleports appear.

IV. Case Study: The Hewlett-Packard Satellite Network

Like most other large companies, HP is extremely interested in finding ways to control its communication costs. With the A.T.&T. divestiture almost complete it seems certain that minimizing communication expenditures will be a more difficult task in the future. After investigating a number of alternatives, HP concluded that a three node network connecting its Cupertino, Boise and Fort Collins facilities had the potential to lower both communication and(with the advent of video teleconferencing) transportation costs. The network consists of three earth stations purchased from Vitalink Communication Corporation. Each earth station is 9.2 meters in diameter and has a maximum bandwidth of 3 Mbps. The earth stations are connected to HP 3000's via high speed V.35 interface. The

Vitalink earth station is fully redundant, providing an extremely high level of reliability. By using both sides of the earth station to communicate with one other node on the network, HP 3000 users can achieve data rates up to 56 Kbps at bit error rates better than one in ten to the seventh. Should a problem develop on one side of the earth station, the other functional side will take control and continue transmitting with no loss of data. Alternatively, HP can use one side of the Cupertino earth station to communicate with Boise and the other side to simultaneously communicate with Fort Collins. By similarly dividing up the Boise and Fort Collins stations, HP can implement a three node mesh network without the need for any multiplexing. When HP adds more nodes the Vitalink Codamux will handle the necessary multiplexing tasks. HP is currently using their satellite network to transmit engineering, manufacturing and financial data amongst the three sites. We have found that this significantly reduces the development cycle of new products as it normally takes a day or two to send a board or chip design from one site to another via U.S. Mail or rather expensive courier packs. By using the satellite network, HP can move gigabytes of data between sites in a few minutes. We will also be able to shorten the financial reporting cycle as sales and shipment data are available in Cupertino the same day they are transmitted from Boise and Fort Collins. All data is, of course, encrypted prior to transmission to prevent interception. In the future, HP will be will be implementing full motion video teleconferencing amongst the three sites. Using compression equipment from Compression Labs Inc.(CLI), HP will be able to send color video, voice and data simultaneously between specially built conference rooms. HP currently spends a great deal of money moving its people between sites on both commercial and private aircraft. By using video teleconferencing we will be able to limit such travel to those instances where a face to face meeting is absolutely necessary, saving time and wear and tear on our people. HP firmly believes that satellite communication is a cost effective method of controlling communication and transportation costs. In the future we will continue to expand our use of the network as the teleconferencing and Direct Broadcast Satellite technologies mature.

Doug McLean is a Product Manager at Hewlett-Packard's Information Networks Division. He has holds an M.B.A. from Dartmouth's Amos Tuck School and B.S. and M.S. degrees in Industrial Engineering from Stanford University. Prior to joining HP, Mr. McLean worked for the ROLM Corporation and the Raytheon Company.

Technical Publication Costs Cut in Half with Laser Printing

Steve Wilk & Sam Boles

Hewlett-Packard

Synopsis

Technical publications enjoy all the standard maladies of other publications: the high cost of type-setting, the rising cost of printing, the long turn-around time from camera-ready copy to multiple copies ready for distribution. But technical publications tend to be unique -- at least in degree -- in their volatility and the urgency of timeliness.

Hewlett-Packard, by being in the computer business, is by definition in the technical publication business. An HP team of development engineers and performance engineers have implemented a methodology using HP3000 document processing facilities and the HP2680 laser printer to cut time costs and dollar costs in half for internal-grade technical publications.

The Problem

Where It All Begins

This story begins in the Performance Center of the Hewlett-Packard computer facility in Cupertino, California. That's where we work as part of the Systems Performance Team running customer benchmarks and doing the performance characterization of many of Hewlett-Packard's hardware and software products.

Big-Ticket Benchmarks . . .

As any of you who've done a benchmark know -- perhaps too well -- they're *expensive*. A complex benchmark emulating a hundred-terminal interactive load across a range of job mixes and configurations can cost *tens of thousands* of dollars -- even *hundreds of thousands*.

. . . and What They Buy

Thru such benchmarks our Systems Performance Team generates some very valuable intelligence on a variety of HP3000 configurations as well as performance optimization techniques.

This intelligence has a flavor to it that is uniquely *real-world*. The benchmarks are run with *customer data* against *customer files* that often are taken directly off a *customer's production* system.

This performance intelligence is also *leading edge*. The benchmarks are often run on newly-released or pre-release products. The environment is such that we can experiment with configurations to calibrate resource sensitivity: changing, running, measuring time after time, validating or rejecting our hypotheses, each time understanding the subtle interplay of system concurrencies a little better - or at least appreciating the *limits of our understanding* a little better.

The environment also enables us to deploy *specialists* on special issues. If it's an ATP timing delay issue or a VIEW screen download issue, we can get the lab expert - perhaps in the next building - to bring to the problem the level of expertise unique to the *author* of the code or the *designer* of the board.

A Central Clearing-House

The Performance Center then is the forum for the early encounters of our products and your world. It has the hardware and software resources, the human resources and the instrumentation and metrics to derive valuable intelligence from these encounters.

So What's The Problem?

The problem is *disseminating* that valuable intelligence. *Sharing* with the HP support organization around the world, and, thru them, the customer base around the world. *Leveraging* the heavy investment in performance characterization of HP hardware and software products. Basically, *spreading the word.*

The Birth of *pn*^{2}

Last year we decided to establish a formal mechanism to communicate our performance information to the field. It's name ended up as *Performance News Notes* -- or *pn*^{2}, as we affectionately nick-named it.

It's a monthly publication, 10 to 20 pages in length (for 1-shot reading). It's current circulation is 2500 readers in the HP organization around the world.

In our original specifications we set out some objectives for the publication:

Quality: It didn't need to be slick and in color, but it did need to be readable and professional in appearance.

Cost: We always aim at zero and compromise to the extent that quality and other necessities dictate.

Speed: It had to be fast. Especially with new product characterization we couldn't tolerate a multiple-day multiple-iteration typesetting cycle, and a 4-day to 2-week printing cycle. But our cost objectives wouldn't allow the premium to expedite these cycles. Also we wanted to use the HP bulk mail system to minimize national and international distribution costs, so labeling and sorting speed were another consideration.

Flexibility: It had to have graphics capability since much of our information is best communicated in graph form. It needed multiple character sets for more intelligible typography.

The Solution -- Set-up Phase

The Laser Printer

We decided to try the *Laser Printer*. It looked like a reasonably good fit for our quality, speed, cost and flexibility objectives. We had an HP2680A on one of the systems in our department, so we gave it a go.

Getting Started with IDS/3000

We spent a few hours getting set up. We used IDSCHAR ("IDS" means Interactive Design System) software and a 2647 graphics terminal to design our logo. That took a couple of hours, counting the tweaking that you're always tempted to do even though *you're the only one* who'll ever notice that one dot out of the 180 per inch that's a little bit out of line.

Then we moved into forms design with IDSFORM software and the 2647. We needed a title page and a mailing label page. This took about 2 hours, plus another hour or 2 of moving things around till everybody -- or almost everybody -- was reasonably happy.

IFS/3000 for Device Specifics

IDSCHAR and IDSFORM generate dot matrices and vectors that are device-independent. This data is then compiled by IFS2680 ("IFS" means Interactive Formatting System), into an ENVIRONMENT file that is compatible with a specific I/O device, eg the HP2680 laser printer. So we used IFS2680 to compile our ENVIRONMENT specifications into an ENVIRONMENT file that the HP3000 could then download to the laser printer.

The ENVIRONMENT file loads character sets (up to 32 different character sets at a time) and forms (up to 32 different forms at a time) into the memory (we have a megabyte of main on our HP2680) that's used by the processor in the laser printer to control the laser so you get an 8-point Helvetica Bold instead of a 10-point Roman Italic. Building the ENVIRONMENT file took another hour or two till we were satisfied.

In all we probably spent about a day getting set up. Some of this time was *learning curve* and some was that "one last touch" of the *amateur artist*, but most was the time necessary to design the form or logo and enter the points and other specs.

The real advantage was the fact that we could do it all ourselves without the start-stop iterations, queues and communications entropy of a graphics department. The initial set up was done in *one continuous uninterrupted block* of time. There were no meetings or key resources to schedule. There was no "now let me see where I was two days ago when I last worked on this" re-think time.

(Of course, the only time you can get that big a block of uninterrupted time is on weekends, so the one-day set up time didn't cost the company anything.)

Typesetting and Composition

We chose TDP (Text and Document Processor) for this because we wanted 2-column capabilities (to enable speed readers to bounce only vertically) and right justification of proportional character sets (to get more balanced "Linotyping"). Also, we needed the capability of including charts from DSG (Decision Support Graphics) integrated with the text.

We included several character sets in our ENVIRONMENT file:

Helvetica Bold: 8-, 14-, 24-point
Helvetica Italic: 8-point
Roman Bold: 10-point
Roman Italic: 10-point
Line Printer: 8-point
Math: 12-point
Inverse Line Printer: 12-point

We specified the appropriate FONTID's to TDP to couple with the ENVIRONMENT file, specified margins, columns, page length, and the like and proceeded to do our typesetting and layout.

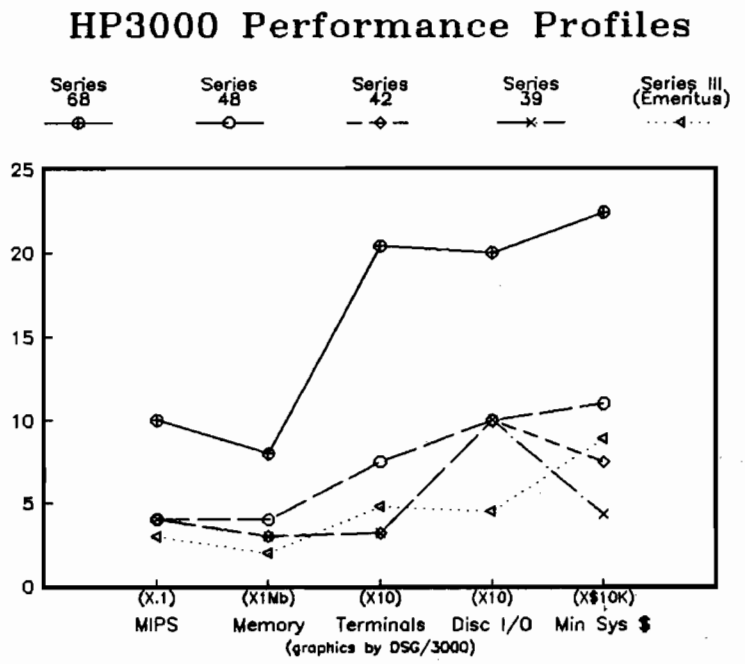
For the layout of charts (bar/line/pie) from DSG and illustrations from HPDRAW, we found that we were doing a lot of trial and error placement, so we set up a template like this:

```
/* ILLUSTRATE rasterfile
/NAME RASTER rasterfile
/ILLUSTRATE figurefile:figure #lines
```

We would NAME the raster file each time we changed the size or the figure (having explicitly purged the old raster file as needed) and therefore had to do another vector-to-raster conversion via the ILLUSTRATE command. Then for the next pass we'd comment out the NAME and second ILLUSTRATE, and de-comment the first ILLUSTRATE so we pulled in the raster form without having to do a vector-to-raster conversion when not needed:

```
/ILLUSTRATE rasterfile
/* NAME RASTER rasterfile
/* ILLUSTRATE figurefile:figure
     #lines
```

This facilitates getting charts like the following integrated with the text:



This template technique we extended to the Issue level, so for the subsequent issues the engineer who had responsibility for that issue could "cookbook" from the archive file of the previous issue without re-inventing the TDP commands. This saves time and gives consistency to the appearance of the publication.

We ran through a number of iterations on our first issue, balancing artwork, policing "widows" (TDP can do a lot of this automatically), cleaning up typo's and again falling into the "one last touch" syndrome of the amateur artist. Once again we found a significant *productivity surge in cutting out the middleman* -- this time the typesetter. We could do a lot of the local

typesetting on the fly as we wrote the articles at the terminal. The global typesetting and layout commands were largely templated (about 75%) from the previous issue. The cut-and-paste we did by computer. And the inevitable last-minute "stop-the-presses" newsflash could be accommodated with only a little pain in a matter of minutes rather than days via the typesetter's queue.

Then finally, there we were with our first issue, camera-ready. We had done the logo and forms design, the typesetting, the graphics and the layout. We were ready for the printer.

We went to our in-house offset service. They were booked solid for the next week and a half. That was 10 days. We couldn't wait that long. If we paid the expedite premium we could get it by the end of next week. That was 5 work days. Better, but still *not good enough.*

What to do?

It was then that someone said, "Why not do it on the laser?"

"The whole thing?" we said.

"Why not? It prints 45 pages a minute."

The next day was Saturday. There's only sporadic use of the laser printer weekends (especially since we had finished our prototype the previous weekend). We needed 1300 copies at 12 pages each. 12 * 1300 = 15,600 pages. 15,600/45 -> 347 mins -> 6 hours.

It looked do-able . . . were we in for an unpleasant surprise.

The Solution -- Manufacturing Phase

We decided to give it a go. That Saturday morning, we set OUTFENCE 10 to prevent interleaving by random print-outs and to enable us to drain the queue as needed but in a controlled fashion. We did a HEADOFF 14 to save paper and reduce the splitting work. We did a TDP FINALQ of our text file to suppress the TDP message at the end. We set COPIES=120 on the SPOOL file to get multiple copies. And we were off and running.

But not for long.

We found that after the first copy, the whole SPOOL file (including the heavy-duty ENVIRONMENT portion) was being down-loaded. Because our ENVIRONMENT file had so much in it, the laser printer was going into a warm-up cycle between each copy. (To conserve energy, the infra-red fusing drum cuts off if not in use for a few seconds, then it has to warm up to the fusing temperature before the printing resumes.)

This cut our print rate to 10 pages a minute. Our 6 hour print job now looked like 25 -- if everything went well. And Mr. Murphy, the silent partner in all such pioneering ventures, said that was not likely to happen. Besides, we were getting an extra page from TDP and that meant every other copy had to be refolded to get the cover on the front. (We used an even number of pages to prevent this but it wasn't working.)

SPOOK to the Rescue

We were just about to hang it up and wait the 10 days for the "real" printer to do the job when someone said, "Why not use the SPOOK APPEND command, build a new SPOOL file with one copy of the ENVIRONMENT and multiple copies of the text. We can even cut out the extra page in the process."

We tried it and it worked. The print speed was up to 35-40 pages a minute -- not far from the 45 ppm rating of the HP2680. We decided to give it another try.

But before we got started, someone said, "How about those name-address labels we were going to print separately on the gummed stock -- why don't we just splice them in while we do the SPOOK APPEND to solve the performance problem?"

Labels on the Fly

We had already tried the TDP MAILER facility to do this. The MAILER is good for multiple addressees for text that does not have a heavy FINAL formatting load, since its formatting performance is about O(n). With 12 pages of 2 column right-justified proportional text plus 8 graphics inserts, we were taking several minutes for a FINAL even with the graphics already in raster format. An O(n) performance would give us 1300 FINAL's at several minutes each. This was not operationally feasible, but a SPOOK APPEND splice might be.

We tried it. The redundant APPEND's (you're APPENDing a text set for each addressee and an ENVIRONMENT set for each 10-15 addressees) builds at a slower (about 2x) rate than your printing rate. If you have enough temporary disc space, you can fire up the APPEND's Friday night, start your printing Saturday morning and have a fairly level work pressure for the operator who's doing the splitting, QA

and boxing for the distribution center. And this does save the time for the label stick-on step.

It wasn't elegant, but the prototype was functionally and performance-wise feasible, so we launched it.

Short on Finesse, but it Worked

Well, with that prototype methodology, we managed to brute-force the first issue in about 18 hours. (The 18 hours includes some cockpit errors I'd rather not talk about. Let's just say that engineers are not necessarily the best operators.) Not exactly the 6 hours we had dreamt of -- but we had beat our alternatives by a week or more and had established that we had the technology and it was just a Small Matter Of Methodology (remember "SMOP" of Large Systems Games fame: Small Matter Of Programming . . . well, this is "SMOM") that needed to be developed.

By the third issue (if you want to find out how short a month is, commit to getting out a monthly publication), we got a bit of prototype software and STREAM commands in place to make our brute force methodology a little more tolerable.

Later, we contracted a 17-year-old student who's working his way thru college doing laser typesetting and printing. Off-loading the grunt work to the student has freed up engineering resources that can be used to get the methodology to an operationally sound condition.

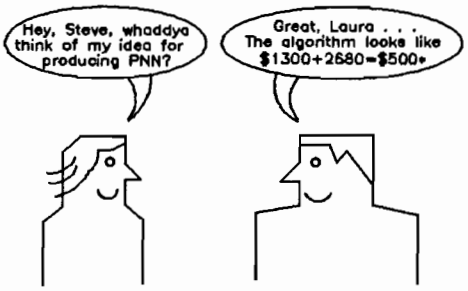
The Results

Structured Development

The approach we've used in evaluating laser technology for typesetting, composition and low/medium volume printing is an extension of Structurism. We're all aware of Structured Programming and Structured Design. This is Structured Development or Prototyping.

We did the first issue with engineers doing the operations, throwing together ad hoc software as necessary to establish that the technology is capable of producing the results. That's established now.

In the next step of development we prototyped some software in BASIC and PASCAL to automate some of the more onerous clerical procedures. The idea here was that if a viable methodology for production were not attainable, we would throw away less code of less value.



(graphics by HPDRAW)

With the crude prototype in place for more than 6 issues, we've seen some interesting results:

12 page performance news notes
2 column right justified proportional
7 fonts, 8 pieces of artwork
(bar and line charts, illustrations)

7:00 PM Friday: last text edit
6:00 AM Monday: 1700 copies printed, pre-sort labeled, split
Variable cost: $500

Variable cost includes labor, paper, toner, carrier, drum cost and maintenance. The machine is idle otherwise,

so the sunk costs of printer depreciation, real estate etc. are excluded.

As word spread on the results of laser typesetting, layout and printing, we applied the methodology to other publications. Here's a recent scenario that may have set a (world-class?) record:

Friday at 3:00 PM, a marketing manager asks if it's possible to have 1000 copies of a 20 page (5 character fonts, 2 column, 1 piece of artwork, no labels) strategy document by Monday (3 calendar days away).

Friday night: Logo design done, forms design started, preliminary environment file compiled.

Saturday: Sample of cover page and first text page built, reviewed, approved. Text data not yet available as ASCII file.

Sunday 11:30 AM: Text data ready for typesetting to start.

Sunday 4:00 PM: Last edits made to text file.

Monday 3:00 AM: Last of 1000 copies printed using 3 laser printers (extras idle on graveyard shift Sunday night.)

We normally call 5:00 PM Friday till 8:00 AM Monday 0 business days. Technically, 11:00 AM Sunday till 3:00 AM Monday is perhaps a turn-around of -1.5 days. In our experience, negative turn-around time is not all that common.

Futures

Our evaluation of the prototype is positive. We've saved at least $5,000 on pn^4 alone, which annualizes at over $10,000.

We are currently evaluating a phase I production version that is operationally sound and simple and may have a 30-50% performance improvement (over the current label splicing methodology.) This would be non-supported contributed library class software.

This same facility may be expandable to include a serial number of each page for sensitive "Do not copy" documents. The control file for this may be the name-address interface file enhanced. This would be non-supported contributed library class software.

Long-term, we see satellite communication to field offices of the "spool file" equivalent and the generation of hard copy as needed at the field location.

Conclusion

The fact that we are continuing to use the prototype version of the laser typesetting/printing methodology is evidence of its economic and functional feasibility. The fact that we are intending to invest the engineering to productionalize the methodology reflects that it has potential for material contribution in the near term.

About the Authors....

Steve Wilk *is the Performance Center Manager at the Hewlett-Packard computer facility in Cupertino, California, with responsibility for performance benchmarks and product characterization. His 18 years in the industry cover a wide range, including programming, systems analysis and program management. With an undergraduate degree in mathematics, Steve received his MBA from the University of San Francisco.*

Sam Boles *is a Systems Engineer in the Systems Performance Center at the Hewlett-Packard computer facility in Cupertino, California. With HP for seven years, Sam's computer experience started back in the AUTOCODER days of the 1401/1410, migrated thru the 360/370 era, and now focuses on HP hardware and software performance characterization. Sam earned his MS at UCLA in Information Systems.*

sebiug12 1015 15Nov83

A DPer's INTRODUCTION TO OFFICE AUTOMATION TRENDS, TECHNOLOGIES, AND APPLICATIONS

Duane Schulz
Hewlett-Packard Company

INTRODUCTION

The MIS Director for the manufacturing division of a Fortune 500 company is returning to his office after grabbing a cup of coffee. When he walks past the Marketing department, which is a big mass mailing system user, he notices two large boxes in the secretaries' area. He asks about these, and discovers that they contain word processing stations, which were purchased from another vendor without his knowledge. When he comments, "we could have handled that for you," he is told that marketing was not very happy with the technical nature of the mailing system, and so did not approach him for help. Shaking his head, he continues on, and is stopped in the hall by one of the group of managers which is participating in a local MBA program. He is asked: "We need some personal computers for our labs in class, and might as well purchase good ones, so we can use them in our jobs, too. Which one should we buy?" When he remarks that all of the capabilities needed for this computation is provided by terminals on his system, and that five languages are available, not to mention the data these managers work with, the requestor backs away, thinking of the general reputation of the MIS group in terms of complicating simple issues.

The MIS Director returns to his office, sits down, and thinks about all he has done in the past few years so that he can start to offer state-of-the-art services. This year, he is planning on implementing capacity planning and shop floor control, financial and market modelling, and other direct uses of the total data base for managers. Yet he has just encountered two indicators that these plans do not really address the desires of the mass of his users. He is discovering that his expertise in data processing does not promise that he can meet the information handling needs of his entire organization. He wonders: "How can I stop this 'creeping office automation' in my company and maintain MIS' position as the information center for the company?"

This paper is based upon the assumption that the primary systems growth which will be encountered in most organizations in the next few years will be in the form of "Office Automation." In order for these systems to truly meet their potential, it is crucial that data processing management is fully cognizant of Office Automation, and able to respond accordingly. If this does not happen, we will continue to see massive growth in the loss of control indicated in the above scenario. I am assuming that the reader is a data processing professional who would like to learn more about Office Automation; my intent is to help these individuals to gain a fuller basic understanding of Office Automation - what it is, why it is important, how it can be used, and what should be done to begin to deal with it. First, it is important to discuss exactly what OA IS.

Many long and academically interesting papers have been published regarding the definition of Office Automation and Office Systems. From the standpoint of a data processing professional, I would like to define Office Automation as follows:

> Office Automation is the introduction of a new technology into an office environment with the intent of resolving a specific problem or class of problems by bringing an individual or group of individuals into contact with this technology.

By this definition, we see many Office Automation (from now on we'll refer to it simply as 'OA') applications which we might not have expected: mailroom equipment, copiers, telephone equipment, and of course, computer systems. This indicates what the future of the office holds, and it helps when a data

processing-oriented person thinks of DP as a part of OA, which it in fact is. If this is denied, either summarily or subtly, the results will likely be similar to those indicated above, and DP may find that it never reaches that "Information Center" status which we all hope for.

Another question which needs answering is "Why is Office Automation important?" The answer is quite clear, and can be given by providing a few facts. Between 1970 and 1980, the US spent $35,000 and 25,000 per worker on agricultural and industrial workers, and achieved 185% and 90% respective productivity gains. In the office, we invested a whopping $2,000 per worker, and were given a 4% productivity increase in return. When you fold in the fact that about 50% of an average organization's payroll goes to office workers, it is clear that OA will grow simply because the Office is the area that is now holding back organizational growth and effectiveness.

Another issue is, of course, technology. Simply stated, in 1960 the ratio of hardware vs. people costs was 50:1. In 1982, it was 1:50 - a complete reversal! This indicates that, due to a technology explosion, rising labor costs, and pent-up demand in office workers, especially managers and "knowledge workers," we WILL see significant investment in OA in the next ten years. This is so obvious that all of the leading 10 computer vendors have indicated that OA will be a top priority for them, and we have begun to see significant growth in computer-based office systems. The danger here is that the customer is a data processing professional, and without a basic understanding of OA, he/she can easily be sold an irrelevant solution by an overzealous vendor. Since our definition above starts with the key word 'technology,' we should start our exploration of OA with a review of the technology involved.

TECHNOLOGY

Office Automation can be seen as a convergence of technologies - in order for an OA solution to be effective, these technologies will need to become more directly related. The illustration below indicates this convergence, and the technology sources which are involved.

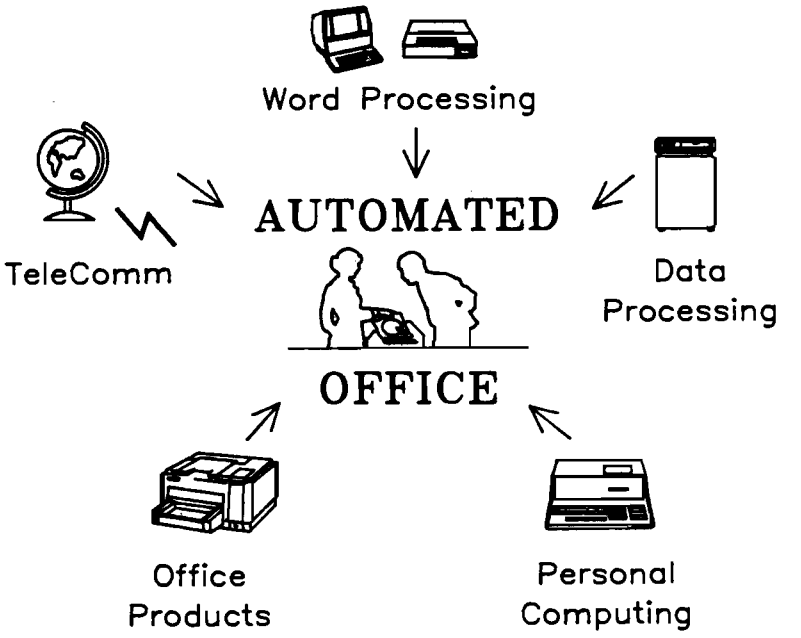


FIGURE 1: TECHNOLOGICAL CONVERGENCE

As you can see (and already know), many groups of vendors are inventing the Office of the Future, and it is important to understand their perspective when evaluating their solution. It is unclear that any vendor with a strong orientation towards any one of these technologies will be able to offer a solution as we will define it later. Proceeding counter-clockwise, let's quickly look at these technologies, and some of the developments in each area.

WORD PROCESSING

Historically, word processing has been provided through the use of specialized, stand-alone or dedicated systems, which has shown to be a long-term misappropriation of revenue. However, in recent years, significant changes have been made by the more notable word processing vendors, outlined below.

WORKSTATIONS. Early WP stations were hardware-based; now most WP workstations are software-based. This allows evolution of the system's capabilities, and also acknowledges that a word processor is actually just a standard microprocessor. We are now seeing the results of this as WP vendors are also providing limited data processing power, on local and central processing units.

PRINTERS. Printer technology has also changed significantly in the past few years, providing enhanced final products at much lower costs and higher reliability. Mechanical devices such as daisy-wheel printers are still heavily used, but laser and ink jet technologies are beginning to provide higher density printing and color output capability, as well as integration of words, graphics, and data processing output. Because of the maintenance required for impact technologies, non-impact technologies, including laser, ink jet, and thermal technologies will become much more prevalent, especially in distributed desktop print workstations.

LOCAL AREA NETWORKS. As a response to the lack of central processors in environments established with stand-alone devices, a facility was needed which would substitute for a central processor in transporting documents from station to station. Using coaxial cable, several vendors began to provide "local area networks." Because of the cable used, extremely high-speed, high-bandwidth communication was available, so LANs began to be explored for multi-media, multi- -vendor transfers, including voice, video, and digital information. The future of LANs is unquestionably bright, but has been clouded by a lack of vendor-independent standards for how they should operate. Now that standards are beginning to appear, such as the IEEE 802.3 standard, the "Information Network" plug we've seen in the XEROX advertisement may become a reality. Because of the high costs involved, it will probably be quite some time before the LAN issue is settled, and many PBX-based local office networks will be put in place as (at least) interim solutions.

TELECOMMUNICATIONS

This area is primarily characterized by the telephone, but we know that it has grown into data-handling in the past few years. Additionally, the reorganization of the Bell System will result in significant and quick growth in this area.

The most basic development in telecommunications is the growth of the carriers which transport a signal, be it voice or data.

Through the implementation of new technologies such as fiber optic cables, digital (DDS) lines and Satellite stations, data integrity and line speeds have increased from faded voices to data transfer rates in excess of 56kB per second.

DIGITAL NETWORKS. In traditional telephone lines, the signal is an analog waveform, and devices are required to modulate and demodulate this signal from digital computer form into analog form, and back again. This is very time-consuming and error prone. The DDS line uses a digital signal to start with, so the only equipment needed is an access device.

SATELLITE STATIONS. Earth-based telephone circuits are limited by the type of wire or cable used to carry the signal (electrons or light beams). Thus, if a circuit from New York to Alaska involved DDS in New York, but old wire-based electromechanical switches in Alaska, the DDS is of limited help. Now, we can use satellites as switches, and remove the wires/cables from most of the circuit. We simply transform the pulse in the circuit into radio signals, send them up to the satellite, then down to the destination, and decode them. The amazing attribute of satellite systems is that, even though the signal travels beyond the atmosphere and back, satellite transmission is much, much faster than earth-based systems.

PBX. As you know, the personal telephone has become as feature-rich as the computer terminal. This is because of the advent of the Private Branch Exchange, or PBX. This facility allows all telephones in an organization to connect to a central control unit, which then connects to the external telephone system. This technology was pioneered by private vendors such as ROLM and Northern Telecomm, but public systems are now becoming available. The significant development in PBX systems was

the acknowledgement that most PBX systems are now controlled by digital mini- or microcomputers. Because of this, PBX systems can handle either voice (analog) or data (digital) signals, and can convert from one form to another. This means that, at minimum, a PBX customer can connect a computer terminal at any telephone plug, given that the PBX system has the ability to interface directly to the computer network. This also means that we will see more and more telephone/CRT/PC workstations, with data and voice capabilities, so that voice messages may be digitized, routed/stored/filed using the computer, and "played back" to another telephone/workstation.

FACSIMILE. To most of us, a FAX device is a box near the copying machine with a rotating drum and a modem which can be used to send a photocopy of a paper document to another office, usually very slowly, and with very low-quality output. Well, the FAX device has also seen the introduction of the microprocessor, and digital FAX stations are now becoming both cheaper and more prevalent. Because the image is digitized before transmission, the speed and quality of output is dramatically improved. Also, the digital nature of the document means that FAX can potentially act as an input device to a PBX or computer-based electronic mail system. Rarely, however, could one "see" the document except through the use of another FAX station, since the document format is not standard or "processed", but rather a matrix of dots.

PUBLIC DATA NETWORKS. An application of some of the carrier technologies shown above is the PDN, also referred to as an X.25 PDN, the name of a network standard. The PDN is a computer-based facility which handles "packets" of information by inserting an address on each packet and passing it through a pool of "nodes" or remote network locations. Eventually, when the recipient "sees" a packet addressed to it, it will be accepted and delivered. The benefit of PDNs, which are offered through private industry, is that large volumes of data can be handled cheaply because dedicated telephone lines to the destination are not needed - they are already part of the network. New locations can "tap" the network as needed. An application of this is found in DS/3000, where a group of HP3000 CPUs (and terminals, if desired) can connect to a PDN for normal DS operations, so that a terminal user in Kansas City might connect via local PDN and run a VPLUS application in Los Angeles, through remote interactive DS operations. The user does not know that all of the screens are being broken into "packets" and rebuilt at the other end.

OFFICE PRODUCTS

Another infrequently-mentioned source of OA technology is the Office Products industry. Again, introduction of microprocessor technology has led to changes in many services provided.

COPIERS. Clearly, the most visible of this class of devices is the copying machine. Two things are happening to these devices: size and intelligence. Micro-based desktop copiers now handle as much work as the traditional "mainframe" copier, and are distributed throughout an office. Because computers are now used to drive many copiers, the ability to connect a digital copier to a network has appeared. An example of this is multi-functional laser printer/copier devices offered by IBM. It is still relatively rare to see small, low cost copiers with RS232 capability, but the new generation of copiers are being called "Intelligent Printer/Copiers," and promise to be a major part of a complete office solution.

MAILROOM SYSTEMS. This is an area which has derived a great deal of benefit from technological advances. Rather than describing these systems, suffice it to say that most mailroom operations can now be accom- plished with the help of digital processors, and that connection of mailroom systems to computer systems are coming soon.

MICROGRAPHICS. Systems involving miniaturization of written information, notably micro-film and -fiche, are becoming incredibly sophisticated. Where we once had a shoot-and-view environment, most complete systems now include computer-based indexing, keywording, and re- trieval, frequently from CRT devices. The records management industry served by micrographics is exploding, and it is becoming difficult to understand when a micrographics system is not an on- line DBMS system. An interesting-to-note technology which shows promise for archiving and training applications is laser disk/ compact audio disk technology, where data, voice, and moving picture storage is available in digital form.

PERSONAL COMPUTING

There is so much going on here that it would be impossible to discuss all OA implications of the personal microprocessor. There are two basic areas where this device is maturing into an OA workstation. First, data communication and software standards are becoming prevalent, so a PC user who is running WORDSTAR on an IBM PC can move a document up to a network processor and distribute it to any user of an MS-DOS-based PC which runs a compatible version of the same software. Secondly, the PC

has begun to sort out applications which are personal and private in nature, and therefore belong on personal computers rather than networks and multi-user systems. These include personal diaries, card files, VISICALC and similar applications. The Personal Computer will without doubt become one of two types of workstation on any complete OA network.

DATA PROCESSING

We all know how much Data Processing has changed in the past 20 years. Basically, there have been four changes which relate directly to Office Automation.

INTERACTIVE COMPUTING. Any system designed to be used by office workers must be an interactive, on-demand system which gives the impression that the user is working on his/her very own computer. Because of the improvements made in interactive computing hardware and software, notably workstations and application generators, data processing systems are viable as central office processors.

MULTI-FUNCTIONALITY. Many vendors, including Hewlett-Packard, understand that the base CPU of a commercial computer system does not necessarily JUST handle data processing. As a result, these on-line systems are being utilized for many of the functions we've seen above. Rather than simply data processing, retrieval and reporting, a good system should be able to handle all aspects of an office's information needs.

NETWORKING. Facilities such as DSN/DS, DSN/IMF, and DSN/PBX indicate that many computer systems vendors have already dealt with the problems of interconnecting interactive workstations in various geographic locations. This is a critical requirement for OA - the system must not simply handle an object, but also provide access to individuals needing this object, regardless of location.

CPU SIZE/POWER. Because of significant research into memory and CPU technologies, relatively powerful data processing machines now occupy the size of a two-drawer filing cabinet, and can be inconspicuously placed in the corner of an office while providing all individuals in that office with OA capabilities.

SOFTWARE/ERGONOMICS. Along with these technological advances, data processing devices are now being provided with novice-oriented software, which is very well-suited to office applications. In the next section, we will look at the basic functions of OA in general; in other words, what can we DO with all of this hardware capability?

OFFICE APPLICATION SOFTWARE

Not surprisingly, understanding the functional intent of most office applications software requires understanding the jobs of the office workers who are asking for it. In fact, if you make a list of the functions performed by and between office workers, including yourself, you will develop the list of classes you will see below! It is surprising how much OA hardware and software is purchased without spending the time to understand the actual work to be performed, and looking at the performance of the equipment/ programs in this context. Basically, office software can be broken into the following functional areas. Programs written to handle one functional area probably do not excel in another, and it is not really very reasonable to expect this. Rather, it is usually better to find software which provides the needed classes of capabilities very well, THEN focus on integrating the different functions for the final office users.

WORD PROCESSING. WP provides users with the ability to create, edit, and print documents. It is very important to match the orientation of each user (technical, managerial, secretarial, casual, dedicated) with a WP facility which matches this orientation.

LIST PROCESSING. This facility allows users to create, modify and maintain "lists", or personal groups of fields/records. A telephone list is a good example of an application for this. Though list management can be provided by DBMS capabilities, List Processors are usually very user-driven and flexible, with little computational power. Lists are frequently merged with word processing documents for mass mailings, etc.

FILING/RETRIEVAL. Though all functions involve "filing", electronic filing and archiving facilities provide the ability to store and find ANY type of object: data, voice, WP, graphics, etc. Keyword retrieval and scanning are frequently-found features.

ELECTRONIC MAIL. This function provides the ability to distribute information from one OA user to any other user or group of users. Like the physical mail system, it should provide the ability to do this by name only, and regardless of information. It should also be able to handle ANY type of information in an OA system, not simply messages.

DECISION SUPPORT. Decision support systems can take many forms, but they include any program which a professional office worker can use to extract and analyze information in the comprehensive data base, providing some specific functions to assist the user in making a decision. This class can therefore include ad-hoc reporting systems, graphics, and modelling systems. One frequently encounters "decision support" systems which provide only retrieval capabilities - unless they provide some analytical FUNCTION, they are not truly decision support systems.

PRESENTATIONS. This includes any facility which allows the user to generate the necessary package to make presentations to others, including flip charts, transparencies, and handouts. A good decision support and word processing package is limited unless another tool is available to "sell" the decision reached using the above tools.

CALENDAR. Meeting scheduling, room scheduling, and personal diary facilities are provided by this type of software. A good package will allow one to attach presentation materials, WP documents, and other notes to an appointment in the system, and handle multiple-person meetings automatically.

MENUING. Given all of the above categories of application, some technique must be employed to remove the user from the operating system, system hardware, and network arrangement. For instance, suppose a user would like to choose between a local WP printer, central laser-based IP/C, or a remote ink jet printer. On an HP3000, this would require a :FILE statement, filling in a menu, and possibly :DSLINE/:REMOTE commands. Using a good menuing facility, the user, who is thinking "pick a printer", would have a menu function which said "PICK A PRINTER". Upon poking this button, the user would see a display of the names of each printer (Betty, Copier, Wichita, etc.), simply pick a name, and the menuing facility would handle the commands and the connection. Since the functions above are discreet tasks (ie: text manipulation is NOT an electronic mail function), a menuing or task management facility is the tool which should provide the link, both functional and cosmetic, needed to "integrate" dissimilar functions from the users' standpoint.

THE HP3000 AS AN O/A NETWORK PROCESSOR

Now that we've looked at the overall hardware and software which will be involved in an OA environment, let's look at the potential functionality of an HP3000 in such an environment. On the next page you will see a chart showing the tasks which can be performed either by or through an HP3000-based network. Rather than discuss this in detail, it is adequate to note that, based upon this diagram, it is clear that the HP3000 and related peripherals, software and networking facilities are functionally more than adequate to act as the basis for a completely automated office.

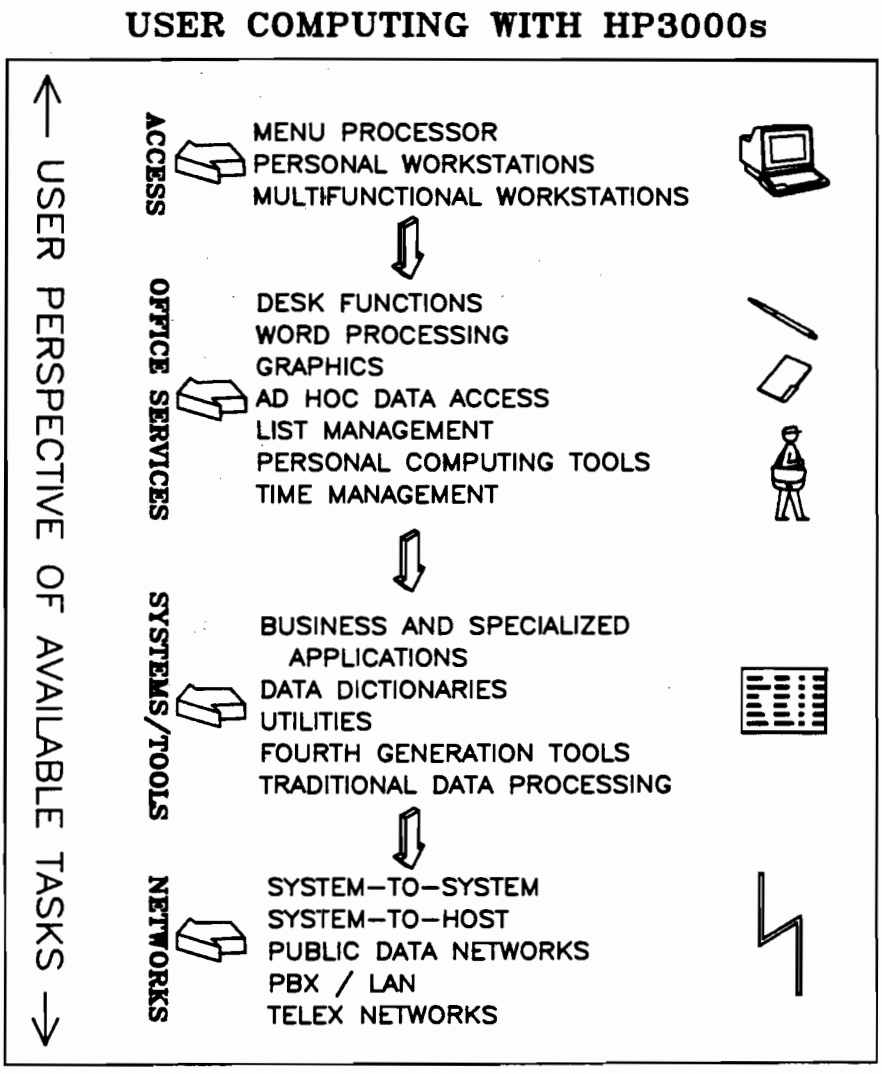


FIGURE 2: HP3000 OFFICE APPLICATION POTENTIAL

Once we know this, another more fundamental - and frequently overlooked - question needs to be answered: how should this capability be applied, and how can we go about identifying these applications?

OFFICE AUTOMATION APPLICATIONS

Suppose our MIS Director decides to deal with his "creeping OA" problem by becoming quite knowledgeable about OA hardware and software, the technology involved, and the trends likely in the next few years. He also works with his vendor to develop a plan for installing the OA hardware and software available from that source over the next 12 months. Finally, he assigns an analyst to interview Marketing department staff and identify information processing needs within this group. After 2 weeks, the analyst returns, saying that he can deal with about 60 percent of the needs of the group, but is embarassed to note that 100 percent of these needs are addressed by the stand-alone word processors.

Data processing analysis works in terms of Information, Processing, and Availability. The analyst in the description above has discovered the difference between data processing and user computing. An entirely new class of users is involved in office automation, and their needs, orientation and expectations are quite different than those involved in production data processing, including user-driven on-line applications.

The most important difference is a focus upon TASK. When a secretary works on a mass mailing, he/she sees is as a series of steps, not a flow of information or processes. Without addressing every single task, the secretary cannot accomplish the mailing When data processing is involved in Office Automation, the proposal all too frequently addresses only a part of the work involved, so office workers see little or no benefit. The assumption of our secretary is that any tool he/she needs to accomplish the mailing (paper, printer, envelopes, telephone, address list, mail cart, etc.) is accessible. In another example, when I sit down to prepare this paper, I need to assume I will have all of the tools I need. Word processing alone is not enough - I need scissors, PressType, graphic arts, etc. to completely perform my task.

Once we begin to focus upon the orientation of office users, it is clear how to identify potential office applications: understand the tasks involved in a person or organization's job. The easiest way to do this is with an interview process - in a high- visibility or complex environment, it might be reasonable to have an OA analyst perform the job right along with the prospective user to fully understand the needs and frustration involved in problem areas. All persons involved in a potential application should be included, especially management and professional staff - if management is not interested in an Office Automation application, the project will likely fail, no matter who is involved.

As specific applications are explored, it is important for data processing management to work with top management in looking at overall company/organizational goals, objectives, etc., to find areas of organizational performance which need work or are desirable targets for Office Automation. In a distribution environment, for instance, excellent customer service might be a basic company objective, and communication with salespersons and sales offices, including order processing, customer service, and credit approval staff might be deteriorating with growth. This would be a good target area because of a clear need, available technology, and overall management concern. Needless to say, it is quite easy to "sell" a project when it involves a concern of top management and a board of directors.

As we have noted, OA applications involve many variables - top management support, technological feasibility, user willingness, a need to address the issue, and a clear benefit which can be measured against at the conclusion of the project. Typically, data processing staff respond to demands to office services by attempting to understand technologies and software attributes, rather than the jobs of the requestors, and the related organizational objectives. This usually leads to difficult relationships, and frequently failure and loss of control. The best position to look for is one of leadership, where a beneficial application can be identified with the cooperation and interest of the user, and problems can be prevented rather than simply met with response.

ELEMENTS OF AN OFFICE SOLUTION

It would be simple to discuss likely OA applications, but we would rather provide more concrete information for use in tackling requests for OA assistance in your organization. To do this, we have identified seven ingredients which most office automation applications have in common. Below is a figure to illustrate these attributes.

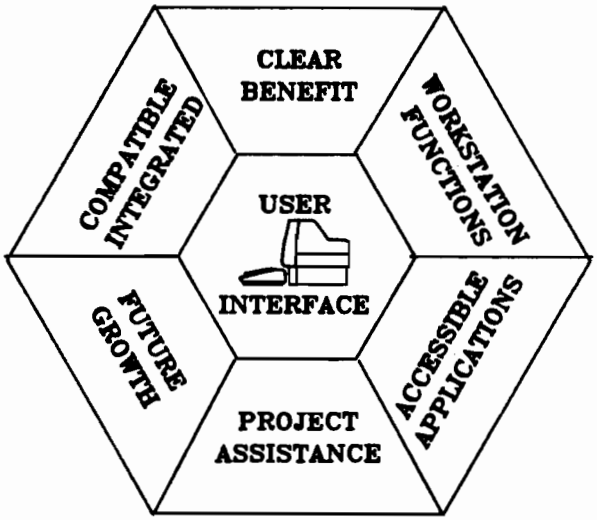


FIGURE 3: ELEMENTS OF AN O/A SOLUTION

USER INTERFACE. Without question, the most important indicator of the success of an OA application is the target user's initial reaction to the way the application looks and feels, or the "user interface". This will determine their willingness to work with it - without a willing participant, all of the following points are irrelevant.

WORKSTATION. In addition to the user interface, the workstation(s) involved should also be thoroughly examined. Will it stay up? Will it handle all potential applications for this user in the next three years, or will it need to be replaced? Can it act as a personal computer? Will it connect to any network available now as well as in the near future? Office workers use only one telephone - they will not accept 2 workstations.

ACCESSIBLE APPLICATIONS. Especially when offering computer-based OA facilities, it is important that their availability is clearly stated and will meet the needs of the user population. I can count on my calendar being available, in my briefcase, 7 days a week, regardless of my location. Therefore, I need (nearly) such access to a computerized calendar. An important detail in providing such accessibility is involvement of operations staff - backup and shutdown procedures will need to be considered. Additionally, system loads must measured, projected and covered BEFORE the application project is started. If these issues are not covered, rumours of poor response and downtime will certainly be encountered. Since - like the telephone - OA applications are frequently close to the pulse of an individual's job, they must remain accessible.

PROJECT ASSISTANCE. Implementation of OA is a project, not a product or program. A vendor who understands this will provide not only the product and training, but also full project assistance for the duration of each pilot group, which usually lasts for 4 to 8 weeks. This is frequently at additional cost, but a vendor who does not take every action necessary to insure your success is simply looking for short-term business. As you see the vendor in this light, the end user sees you in the same light. You will need to handle all aspects of the user's initial implementations of the product. We will discuss these requirements further below.

FUTURE GROWTH. OA applications are usually steps in completely automating the office, whether this is planned or not. If a word processor is purchased to prepare internal memos and personnel-related materials, can it later be used to transport these documents to all possible recipients using an electronic mail system? Will they be readable by users without a WP station or terminal? Again, only a multi-functional system capable of evolutionary application growth is sensible given the quickly-changing nature of today's DP and OA community. It is quite common to see users

wish to solve a short term problem while creating a long term problem, and data processing and MIS individuals can help to avoid this.

COMPATIBLE/INTEGRATED. If a solution such as the one above is implemented, and an electronic mail function is employed, how is it integrated into the user's menu? How many steps are required to move the WP document into the mail system and make it readable? A helpful exercise in this area is to make a list of all "objects" (you presently call them files...) your final OA system will handle. Across the page, indicate each "function" (WP, EMail, etc.) which will be offered in the complete system. In this matrix, indicate how each function needs to use each object (create/edit, print, file, include, etc.) - this will help to qualify vendors, and insure that users' needs have been thoroughly identified. Nothing can be more frustrating than spending $100,000 on a complete office system, simply to find a user with a pair of scissors and tape, performing functions which will not be able to be shared throughout the system.

CLEAR BENEFIT. Though this has been mentioned before, it is absolutely critical to the success of new OA applications to state the planned benefit of the application, and have management, DP, and office users of this application agree that this is the desired result. All too many OA applications have been implemented without doing this, and even when they are successful, it is usually because they are technically intriguing. Finally, how will this benefit help to achieve a higher level organizational objective?

O/A : IMPLICATIONS FOR DATA PROCESSING

Office Automation applications indicate some impact on existing Data Processing organizations which are involved in these solutions. Not surprisingly, they tend to fall into the categories of technical issues and support issues.

TECHNICAL IMPACT

Four technical issues are involved here. First, the system will now be used to "handle objects" rather than "process information", and these objects will need to be managed. Who will archive, retrieve, and perform housekeeping on these objects? Frequently, electronic filing and retrieval is needed on top of the original user application.

Next, how will the devices involved (printers, plotters, PC's, etc.) be made available, and who will support them? Will you need mini-courses on troubleshooting for end users? It is very common to see WP users circled around a malfunctioning printer with no idea of how to isolate the problem, and no help available.

Given the idea of multiple types of objects indicated above, how will these objects need to be integrated for the user? Here we find a need for a "task/menu administrator" who can translate the system's technical needs into a button for a user to push. Again, all too frequently, we find DP personnel trying to teach administrative users how to user FCOPY rather than establishing a conversational menu function for the user.

Finally, will the system(s) used be able to support the application? If the solution is computer-based, CPU, I/O, and memory resources will be needed, and this need should be projected and handled. Again, "poor response" might better be called "poor system management". These things can be resolved without the user's knowledge that a central system is involved - this is not related to his/her office task.

SUPPORT IMPLICATIONS

In supporting office users, we cannot overemphasize the importance of providing non-technical support to these users. In most data processing organizations, this cannot presently be offered - no matter how "people-oriented" a programmer/analyst is, he/she will still have difficulty in being accepted by secretarial and managerial users during an OA project. The traditional support lines for data processing users are rarely sufficient to provide the necessary response and empathy needed for the personalities involved in office automation. Because of this, I recommend that data processing organizations establish an "office products coordinator", an individual who will provide ALL first-line contact for office applications, including training, user meetings, day- to-day support, interviews, etc. This person would then work directly with data processing personnel who have responsibility for supporting the OA software and hardware. Additionally, this individual should have direct access to OA vendor support.

Because of the significant differences in supporting an OA environment, this new focal point is needed to further acceptance of the

project and improve communications. A secretary will accept information from another secretary much more quickly than from a data processing professional, regardless of their orientation and demeanor. A possible support arrangement which includes this office support individual is shown below.

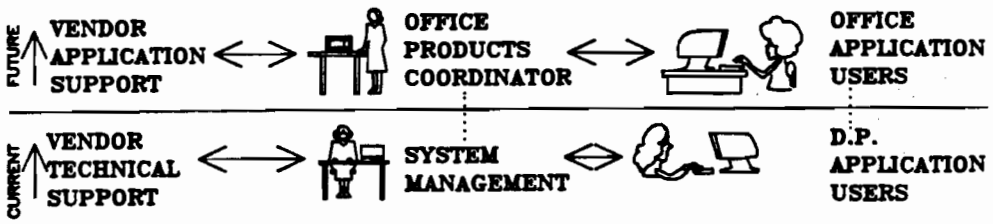


FIGURE 4: O/A SUPPORT IMPLICATIONS

CONCLUSION

Perhaps the best way to conclude an overview such as this is to suggest some activities which will help a data processing professional to become more involved in successful OA systems, and stay in the center of information processing for their organization.

First, start reading about office automation - there are several very good trade magazines and professional societies. Administrative managers or existing OA users in your organization can help you to find out more about these. This will help you to keep abreast of new developments, as well as become comfortable with the vocabulary of OA.

Next, determine what your current vendor(s) can do in OA, and explore how implementation of O/A might alter your future plans in data processing. While you are doing this, it would be helpful to conduct an informal survey within your organization to see where Office Automation would be of benefit, who would be interested in exploring OA, and who can afford an office solution. Further, it might be worthwhile to use the information you gather to conduct an "Office Automation Fair" using the facilities your vendor indicates are now available.

Finally, acting as a leading force in exploring OA, develop a short- and long-term plans for implementing Office Automation solutions. If your group already has an OA committee, join it and help them to develop this plan. Acting from a plan can save a great deal of time and money by implementing OA only where it can help, rather than simply where you are told to implement it.

One of the most interesting attributes of Office Automation from a data processing standpoint is that OA systems require very little technical effort to implement. For instance, a startup data processing organization may take 3 months to begin its first data processing application, but electronic mail can be available within hours of the initial startup of the system, and graphics can be running and used within 15 minutes by non-DP users! In the end, the most evident benefit of computer-based Office Automation is as a natural step in the evolution of user computing. Given the visibility and enjoyment it provides to any data processing installation, it remains a mystery why every HP3000 installation is not involved in Office Automation.

Duane Schulz Hewlett-Packard Company

Duane Schulz joined Hewlett-Packard in 1980 with nearly 10 years of experience in managing small data processing installations, in service, manufacturing, retailing, and distribution environments. As an HP customer, he began to implement and work with Office Automation applications.

Since joining HP as a Systems Engineer, Duane has become an Application Specialist for Office Automation, and has been involved in factory and field activities related to the development of HP office systems. He is an active participant in the HP3000 IUG, and speaks to groups of Office Automation and Data Processing professionals regularly.

Which of HP's Many Word-Processing Systems is for You? A Practical Point of View

Chuck Thompson
Systems Engineer
Hewlett-Packard (NZ) Ltd.

Are you in the market for a word-processing system? You may be thinking about starting up an office automation scheme, and word processing is the first logical step. There are many systems available on the market designed to do just word processing, and do it very well. Since you already own Hewlett Packard equipment, however, you would like to stay with the same manufacturer. Let me say at the outset here, that Hewlet-Packard word processing systems are not designed stand alone as word processors, but are supposed to take a significant part in a much larger picture: integrated office automation. Hence, HP's word processing solutions are, in most cases, the best for companies that need word processing to run alongised other applications on the same set of hardware.

HP has a variety of tools that can be used for word processing and there are a few third-party systems available that work well on HP machines. Technically, some of the software packages I'll be talking about today aren't really word pcoessors, but mahy customers use them just for that purpose. The main reason is the significantly lower cost. HP's word processing software consists of these six products (as of November 20, 1983):

Editor/3000	The basic line editor that comes with the Fundamental Operating System
TDP/3000	The text and document processor
HPSlate	A screen oriented editor and word-processor with capabilities similar to TDP's
HPWord	HP's most complete word-processing system
Word/100	A full-screen word processor for the HP120 and 125 (also known ans Spellbinder)
WordStar/100	The most capable word processor available on all the Series 100 computers (HP120, 125, and 150)

Editor/3000

This is one of those pseudo-word-processors I mentioned before. All 3000 users are familiar with this tool, as it comes bundled with MPE. This means that it is the cheapest possible editor available on the 3000. This alone can be the chief reason it is considered when a word processor is being sought. I don't know of anybody who uses this as their word processor, but in theory at least, it can be useful.

Advantages

- Uses straight ASCII "flat files," so can be used to edit files from virtually any software.
- can be very powerful for accomplished users.

- doesn't require the purchase of special software or hardware.

Disadvantages

- Command-driven (difficult to learn).
- line-oriented (no screen handling).
- cumbersome and slow to edit odd-sized files.
- No capabilities for output formatting (must be done manually).

TDP/3000

Very similar to Editor/3000, TDP is a "text and document processor." It has a limited "screen mode" which allows the user to take advantage of the basic screen-editing capabilities of a block-mode terminal (any terminal except the 1205, 125, and 2621), and adds many special capabilities for document processing, scuh as automatic Table of Contents and Index generation. TDP is not normally used as a company's only word-processor, but has definite advantages over other word-processing systems when it comes to producing long documents.

Advantages

- Works on ASCII "flat" files
- Very powerful command language
- full-screen capability
- very powerful text formatting capabilities
- built-in calculator
- hyphenation dictionary for formatting
- mass mailing and macro capability

Disadvantages

- command driven
- line-oriented
- screen editor is very limited
- no use of function keys
- many commands different from Editor/3000 so learning can take longer
- relatively expensive (about $6,300.00) (all prices, by the way, are as of 20 November 83, in US$, factory base)

HPSlate

This is the lowest-cost "real" word-processing system available from HP. It is a full-screen editor only, and runs (or is supposed to run) on any true HP block-mode terminal (in other words, the 120/125 are not supported, even when running the block/format emulation software). Slate relies heavily on the capabilities of the terminal, especially terminal memory, but terminals with small internal memories like the 2382 (Shadow) terminal can run in "1/2 screen" mode. This means that only half the current page is displayed at one time. The normal editing features of block mode terminals (insert/delete character, etc.) are used in Slate, but the software also adds a full set of function keys to do other word-processing tasks, like justifying a page, or underlining. Some things in HPSlate are very much different from other word-processors because of the way the terminal is used to do most of the processing, yet most terminals will work. For example, only a carriage return sends your information to the computer, so no automatic word-wrap is done. Also, a page is an entity in Slate, since that's how much a terminal can hold in its memory. This means that no pagination need be done, yet text must manually be moved from page to page. Slate allows the user to access the TDP formatter via softkeys (that is, if you have TDP), and uses a VPLUS-like menu for control of printer output. The use of softkeys throughout, and the very comprehensive help subsystem, make HPSlate very user-friendly, but the product still assumes that the user knows what he or she is doing.

Advantages:

- full screen
- includes standard word-processing functions
- works on any block mode terminal
- uses function keys extensively
- excellent help facility, including a general overview
- allows access to TDP, Typist, and MPE commands
- keeps text in compressed files to save on disc space
- comparatively inexpensive (about $3000.00)

Disadvantages:

- moving text between page can be difficult; does not allow movement of "page marker" within text.
- no automatic word wraparound (at the end of a line)
- no automatic page wraparound (must manually call up a blank page)

- must convert to/from Slate format in order to use ASCII files
- can cause high CPU utilization, since it doesn't assume much intelligence in the terminal

HPWord

This is the largest, most powerful word processing system offered by HP to run on the 3000. It incorporates all the facilities of most stand-alone word-processors made by our competitors, and adds the capability of integration with other office automation software: HPWord files are readable by HPDeskManager, and can contain figures or charts created with DSG, EasyChart, or HPDraw. HPWord requires a special terminal, either a 2626W, a 2625A with the HPWord option, or the 2628A. Most of the word-processing function, though, is downloaded to minimized. HPWord is often used as a complete operating environment, from file (document) handling to printing control. Spooling to letter quality printers, like to HP 2601, is handled by HPWord, allowing up to eight workstations to use one printer. HPWord also has the capabilities of assigning environments to output documents destined for letter quality printers or for laser printers.

HPWord makes extensive use of function keys and specially labelled keys. This makes the learning time very short, but the lack of a help facility makes it a little more difficult to rememeber what you learned. This system, unlike the previously mentioned software, is designed to be used mainly by secretaries and data-entry/word-processing people. In my experience, the only people who ever really know HPWord are those who use it. The data processing staff and system managers never seem to need to know more about it than how to change the terminal and printer configuration.

Advantages:

- full function, powerful word-processor
- integrated into HP's family of interactive office software products
- user-friendly; designed for non-DP personnel
- most processing is done in the terminal, leaving the CPU free for other tasks
- extensive use of function keys and specially-labelled keys
- all enhancements (bold, underline, italics) shown on the screen as they would appear on the printer
- built-in calculator and mass mailing functions
- accomodates wide documents by scrolling right and left as well as up and down
- excellent self-paced and classroom training available

Disadvantages:

- requires special terminal
- requires system resources for spooling to letter-quality printers
- no automatic pagination; this means there is one more step needed to print out a document.
- keeps documents in a special format, readable only from HPWord or HPDeskManager
- left and right justified text is not shown on the screen
- conversion to ASCII is combersome
- no help facility
- no access to MPE commands (could be an advantage)
- could be expensive (about $5000.00 for software, plus about $3200.00 per 2628A terminal, and $3500.00 per 2601 printer)
- limited document processing capabilities; could, however, be used in conjunction with TDP

Word/100

This was the first word-processing software available on the Series 100 personal computers, the HP 120 and 125. It is available on other CP/M-based personal computers as Spellbinder. It uses the full capabilities of the screen and adds function keys for most commands. This software operates on a command-mode/edit-mode system, which

allows the user interface to distinguish between the two systems. All the screen-oriented software I've discussed so far uses function keys and specially-labelled editing keys. Word/100 only used function keys as a way of issuing commands while in command mode. These modes can be confusing when you forget which mode you are in.

Word/100 includes a macro definition capability which is extremely powerful, and includes some pre-defined macros for doing such things as mass mailings, multi-column printing, and sorting. Character enhancements (underlining, bold printing, etc.) are shown on the screen in inverse video. This makes Word/100 a powerful editing tool for users of personal computers, but the problems with switching between edit and command mode make it difficult to learn for non-computer users.

Advantages:

- stand alone on a personal computer; does not need a central computer system
- powerful yet simple
- screen-oriented
- makes use of function keys and terminal editing capabilities
- shows enhancements in inverse video
- allows easy conversion to ASCII flat files for transfer to the 3000
- can be very inexpensive; often given away with a Series 100 system

Disadvantages:

- must run on an HP 120 or 125
- very confusing and difficult for new and casual users and non-DP personnel
- limited fix or enhancemet support from HP
- must purchase a separate copy fro each system it's used on
- poor help facility

WordStar/100

This is probably the most popular, and the most reliable, word-processing software available on personal computers. I'll even go so far as to say that it is the best-written software I've seen on personal computers. In its non-HP form, it uses only control keys for its commands, getting rid of any edit/command mode problems. HP has taken the standard woftware one step further and allowed the use of function keys for almost every command. This makes it extremely easy for a new or casual user to create and edit documents, but the process can be slowed down by searching through the numerous sets of function keys for the one you want. This latter problem is solved quite easily, though - the control key commands still work, so that you can use them once you get familiar with the editing process.

WordStar includes two add-on packages to increase its power: MailMerge, a mass mailing program, and SpellStar, a word-by-word (context ignorant) spelling checker. These combine to give WordStar almost the power of Word/100, with much greater ease of use. SpellStar allows the creation of user-specific secondary dictionaries, as well as adding to the standard 27000-word main dictions, like operator prompts for words, pauses for ribbon and paper changes, and operator instructions, as well as an easy to use mass mailing program.

Advantages:

- full screen, full function, character-mode word-processor
- well integrated with most other Series 100 software
- includes HPTouch support on the HP 150
- function key driven
- easy to learn for non-DP people
- optional spelling checker
- optional mass mailing capability
- accepts ASCII flat files produced on other software
- word wrap can be turned on or off at will
- scrolls sideways to facilitate editing of wide files
- stand-alone on a personal computer - no main CPU to drag down
- allows for micro-justification capability of HP 2601

Disadvantages:

- impossible to create real ASCII files from documents
- needs a dedicated personal computer
- very expensive compared to Word/100 (about $500.00 for WordStar, plus $250.00 for SpellStar and $150.00 for MailMerge), but is sometimes given away with 120 or 125 systems
- requires a separate copy fro each system it will be used on
- help facility is good for HP people, unintelligible to non-DP users, although function key labels are excellent

Conclusions

I'll not dwell too much more on the first two systems mentioned, except to say that they, and other third-party editors, like QEDIT, can be used to fulfill word-processing roles if the cost of other software precludes the purchase of proper word-processors.

The rest of the systems fall easily into two categories: those that run on the 3000 and those that run on our personal computers. Choosing between these two should be a major decision, since each has definite benefits over the other. Thought should be put into what your word-processing load will be, whether you want to put the extra load on the 3000 (or conversely, whether you want your WP staff to fight with other users of the mainframe), as well as the equipment you currently own. If you already own a personal computer, you might consider using it for word-processing as well, and if you already own a 3000, you might consider buying the WP terminals needed. I have seen both methods used su ccessfully in environments where the main users are non-DP "secretarial" staff. The 3000-based system works well where all the temrinals are in close proximity to the mainframe (remember that 50-foot limit on RS232 communication), mostly in a room dedicated to word processing. The personal computer system is useful in a situation where the machines are places anywhere in a large building - they may even be on trolleys for easy movement - and access is not needed to the mainframe. Another thing to think about is that 3000 files can be as big as you want, while files on the Series 100 are constrained by the size of the discs you buy.

The next smaller division is obvious: the decision between HPSlate and HPWord on the 3000 and between Word/100 and WordStar on the Series 100. The former really boils down to whether chiefly WP staff are going to be doing word- processing, or whether the main users will be end users and DP personnel, who won't necessarily have word-processing terminals. The question of using Word/10 0 or WordStar can be answered by looking again at who will be using the software. Almost always, the answer will be WordStar, chiefly because of the ease of use a nd the SpellStar software. Notice that HP has chosen not to carry the Word/100 software over to the 150. This should tell you something about our commitment to user friendliness.

One last thought: once the final decision is made, you're not really stuck with it, since most of these software packages created documents that can be transported (through ASCII files) to other software packages. Usually, though, enahncements and page headings and footings are not translatable, so some work will still go into system conversion.

Getting Started in Data Capture

Bruce Toback
Infotek Systems

I. Introduction

Over the past decade, on-line computer usage has been replacing traditional batch usage at an increasing rate. The reasons for this transformation are many, but most center around responsiveness: the ability to use the computer as an information source in real time, asking questions which require the most current answers.

In point of fact, however, a large portion of this transformation has been the replacement of batch systems with "faster" batch systems. Data are still collected and recorded manually, then sent to a terminal operator for entry into the "on-line" system. This method has advantages over traditional batch systems in that data are made available sooner than with conventional batch processing and inquiry is easier, but responsiveness can still suffer. An additional consideration is the accuracy of information presented to the computer. In both traditional batch systems and "on-line" batch systems, information is copied at least twice and perhaps several times on its way to the computer. Each translation increases the chance for error.

More recently, source data capture has been introduced. The idea of source data capture is not new, of course: every mechanical cash register is a source data capture facility, recording sales transaction information as each transaction occurs. Using source data capture as input to a computer, however, is a relatively recent innovation. Data capture has been in use quite extensively in the retail sales field in the form of point-of-sale terminals and is now being used in manufacturing as well.

This paper is intended as an introduction to data capture. It will serve as a guide during a data capture project definition, and provide a list of ideas and issues to focus on as you design your data capture system.

II. When data capture is used

In general, data capture is useful in a manufacturing environment when at least one of the following is true:

- The information needed is highly volatile (i.e., changes frequently)
- The information needed is too voluminous to be recorded and keypunched in a timely manner
- The information required is already available in machine readable form as a by-product of an existing manufacturing operation.

Note that in general, source data capture means only that information is recorded electronically as soon as it is generated. This may mean that electronically-assisted means exist to capture the requrired information or may simply mean that the person creating the information (e.g., the sales clerk or stock clerk) enters the information by hand into a suitable terminal.

Let us examine each of these criteria in detail. First, information may be considered volatile when its lifetime is shorter than the turnaround time of the system used to record it. For example, on a fast-moving shop floor,

a given assembly may move in processing from operation to operation several times in one day. If information is entered and recorded once a day, the report will be accurate only for those assemblies that are stuck. This kind of reporting is usually of limited value.

Information which is too voluminous to be keypunched normally can include such things as point-of-sale inventory transactions, e.g., grocery checkout; kit pull information when lot tracking is in use; and of course, job tracking information on a busy shop floor. In general, the volatility of information is related to its volume: information which consists of only a few data items can usually be entered into a data processing system in time to be reported before the end of its useful life.

Finally, information which may already be available in machine readable form includes such things as, again, grocery checkout, where UPC or EAN codes are pre-stamped on packages by the manufacturer; electronic time clocks which can record time and attendance information at the same time that an employee's time card is being punched; and electronic test equipment which may be computer controlled. In some cases, as in retail checkout, the machine-readable coding may be inherent in the operation being tracked by the computer. In other cases, such coding may be added at neglegible cost, such as when turnaround documents are already produced by computer.

III. Defining your Data Capture Needs

Once you have determined that your data entry needs can be met by a data capture system, your next task will be to design the system. The design can be broken down into three main concerns:

- o Volume capabilities
- o Volatility problems
- o Using preexisting data

Volume Capabilities

In order to design a data capture system, an upper limit on transaction volume must be determined. How this is done depends on the type of data that you intend to capture, and on what capabilities already exist for capturing it. When converting from an "on-line batch" system, the best indicator may be the number of turn-around documents currently being keyed into the system, multiplied by the number of transactions recorded on each document. This last is very important since in a properly designed data capture system, each "real" transaction (e.g., item sold, job move, etc.) should equate to one computer transaction.

If no automated data processing facility is currently in use to record the information you are trying to capture, then the volume should be estimated by the personnel most involved with the data.

Number of terminals Once you have determined the total transaction volume for each function you wish to perform, you will need to find the number of terminals required to support that volume. The number of terminals required depends both on the total transaction volume and the number of types of transactions.

Transaction types are important becase each function may take a different amount of time to perform, and more importantly, may take place at a different location from other functions. In general, enough terminals to support the volume for a particular function should be present in the immediate area of transaction generation. This is self-evident when machine-readable information is input through the terminals, but may not be so evident when operators are required to key in information. If an operator needs to "go over to" a terminal to make an input, the tendency will be for the operator to batch input and do several transactions at one time. This may result in both translation problems as operators write down transactions for later entry, or in timing problems as operators delay entering important information until a break period.

For hand data entry, approximately five seconds are required for a 10-digit numeric entry. This may be reduced substantially by using a machine-readable code such as bar code or OCR.

Machine-readable input Machine-readable input can have a significant effect on the volume capabilities and terminal requirements of your data capture system. First, of course, data entry speed can be increased dramatically because

operator keying is no longer required. Bar code or OCR scanning can reduce the operation of entering a 10-digit work order number to a single wand stroke, with almost no possibility of keying errors. However, a "one-operator, one-station" arrangement becomes almost mandatory, since sharing a "personal" device such as a bar code wand is generally reported as difficult. If information is coming from another computer, as in the case of automatic test result reporting, the system must be by definition one-operator, one station.

Volatility Problems

If the data you are trying to capture is highly volatile, you will need to consider several additional factors. Most of these are treated as application design problems, but will need to be considered during your system definition.

System availability Availability of the computer system is of interest in even a batch system, but is much more important in an on-line system. Availability is critical to the success of a data capture system dealing with volatile data. Imagine for a moment the chaos that might result from a malfunction of a centralized electronic door lock control! The data in this case are extremely volatile - someone wants in (or out) now - and it is gone forever if it is not captured at the instant it is generated. If your data are this volatile, you should make provision for spare or backup terminals, and for rapid switchover to a backup computer if practical.

In a situation involving volatile data, terminal reliability is as important as central system reliability. If an operator loses access to the system even though the system is still operational, data from that operator will be batched or lost, or the operator will simply not do any work until access is restored.

Many of the schemes for connecting data capture terminals to the HP3000 involve shared equipment, leading to the possibility of single-point system failure. For example, if all terminals are on a single multipoint line, failure of the INP or SSLC, of the factory data link adaptor (3074A), the cable itself, or certain terminal failures can render the entire data capture network inoperable. Suitable backup precautions must be taken, e.g., splitting the network into two or more lines, or connecting critical terminals point-to-point so that failure of an ATP, ATC, or ADCC can be worked around rapidly by moving the terminal to a port on a different ATP, ATC, or ADCC. The net effect of these considerations is that you should consider the possibility of using redundant hardware in situations in which data are extremely volatile.

Data Recovery In addition to high data collection equipment uptime, volatile data collection requires that you consider recovery methodologies in the event of system failures. These can take several forms. Assuming that retention of paper backup is undesirable, or that no paper backup exists to begin with, the most obvious recovery technique is user logging. This service, provided at an elementary level by MPE and at a more sophisticated level by IMAGE, causes MPE to attempt to journalize every file update to tape. The implication, however, is that you are willing to dedicate a serial medium, either tape, disc, or cartridge tape, to data logging whenever the data capture system is active. You should plan for this in determining your hardware requirements.

In addition to hardware requirements for journalization, you will need to examine your backup strategy for compatibility with logging. In particular, if you are using IMAGE, you should use DBSTORE instead of STORE or SYSDUMP to back up. This is because the Hewlett-Packard logging/recovery utility, DBRECOV, will examine the last **DBSTORE** date to determine whether your transaction log and data base "match" for recovery. If you normally do long-period incremental dumps (i.e., every two or three days), you may in addition be faced with a considerable recovery period after a system crash. While this is preferable to losing several days' worth of transactions (!), it also causes a net decrease in system availability as perceived by your data capture operators.

Finally, if you are not using IMAGE, you will need to design your own recovery utilities for your data base. If you implement your data base with KSAM or with a third-party data base system, this may be an important planning consideration.

Machine-readable documents If you choose to use paper backup instead of transaction logging, you should seriously consider the use of machine-readable turnaround documents. This will facilitate the recovery process in the event of a system failure, since a large amount of keypunching will not need to be redone. Rekeying already-entered data can be very destructive to the morale of an operator to whom computer input may be an unrewarding burden in any case (see User Training, below).

Using Machine-Readable Data

If you have chosen to use data capture to take advantage of existing machine-readable data, you will need to consider exactly how you will read the machine-readable information.

Bar coding Bar coding is becoming extremely popular in manufacturing as well as retail environments. It is easy and inexpensive to generate (and may in fact already exist, as in the case of the UPC/EAN codes in the retail field), easily read either under manual control or automatically by non-contact scanners, and is very reliable. Bar code readers are made by a large number of manufacturers including (in the United States) Interface Mechanisms (Intermec), Hewlett-Packard, Burr-Brown, Epic Data, and others. Some of these companies in addition manufcture complete data capture subsystems for inclusion on an existing computer system.

Bar codes may or may not be human-readable in addition to being machine-readable. Some bar code standards (UPC and EAN, for example) stipulate that the code shall have a human-readable translation above, under, or beside the code itself; you can, of course, design your system to provide the translation.

Optical Character Recognition Less popular than bar code is a system called Optical Character Recognition. Specially printed labels containing ordinary numerals (and sometimes letters as well) are read by a hand-held contact scanner. While the resulting label is smaller than a bar code label since the optical characters are typically more dense do not need translation, read reliability is rather poor with most systems, and several tries may be needed to read a label. OCR systems are usually manufactured as turnkey add-ons to existing computer systems. Some manufacturers of point-of-sale terminals offer OCR as an option.

Automated Test Equipment Automated test equipment systems (ATE) are becoming more popular as prices of versatile systems have dropped dramatically. By combining an ATE system with bar code or some other means of identifying assemblies, a completely automated testing **and** inspection records system may be set up. Interface to the HP3000 may be directly through RS-232C or RS-422 (terminal interfaces), or through GPIB or ordinary parallel interfaces using the HP3078 data collection terminal.

IV. Design and Implementation

System design aspects and their corresponding implementation aspects may be broken down into four areas:

- o Topology
- o User Interface
- o Application System
- o User Training

Topology

The network topology that you choose to implement depends on your needs as discussed earlier. The key questions are:

- o Where is the data available?
- o What form does it take?
- o Who has the data?
- o How will the data be communicated?

Where? On the principle that terminals should be located as close as possible to the source of the information that will be entered through them, you will need to locate as precisely as possible the locations in your plant which generate the information that your system is intended to capture. While this seems self-evident, all too often a location is denied a terminal because the usage would be low, even though the information generated there is as volatile as at higher volume locations. The result, of course, is "batching" of input and concomitant data timing problems.

Data capture terminals can usually be placed in crowded areas by using wall mounting. Hewlett-Packard's 3076A terminal is especially designed for permanent wall mounting; its keyboard and display may be used easily while sitting or standing. Wall mounting is also convenient when information will be entered by operators while walking or driving, as in access control or timekeeping applications.

What? The data to be entered can take many different forms. In many instances, data starts simply as knowledge: the receiving clerk knows that something has come in from a vendor, for example. In other cases, the data is part of an existing document, e.g., the packing slip contains a packing slip number that must be entered. If the data consists simply of knowledge, it is usually most expedient to simply key information into a convenient terminal. If data are contained on a preexisting document, it may be possible to code the document for machine reading. This is especially true if the preexisting document is generated within the company: the document can be generated on a printer capable of printing bar code to be read in later.

If the data to be captured exists in another computer (an ATE system, for example), no conversion or keying will be necessary.

Who? The data to be captured should be entered by the person (or object) most closely associated with it. In terms of the organization, the person or political entity that is held responsible for the accuracy, completeness, and timeliness of the tracked activity should have the responsibility for data entry. This means that terminals will be located close to the people handling the data: in the stockroom, on the shop floor, at the receiving dock, and so on. Where several people have knowledge of a particular physical event, the person who should be responsible for data entry should be the one who must have the knowledge for the business to run smoothly.

As an example, consider the shop floor controller moving assemblies on a shop floor. During normal operations, two people have the knowledge that an assembly has been moved to a new workstation. The shop floor controller, of course, is aware of it, but the because he or she caused the physical event. However, the assembly must be "formally" entered in the queue at the new workcenter. Thus, the workcenter's supervisor or lead operator, if any, should be responsible for data entry: the data capture system will then show the event only after action can be taken on it.

In some cases information will be subject to audit, and it may be desirable to limit update access. This may be accomplished with a machine-readable ID such as a magnetic striped badge or laminated bar code. In such cases, it may be expedient to use a method of physical identification similar to that which is used for automated data entry.

How? Once an event has taken place, is discerned to have done so by somebody, and a record of the event has been made and entered, the record must be transmitted to the computer. This is done either directly by the terminal, or by a data capture subsystem to which the terminal is connected. Hewlett-Packard, in general, favors the former approach; most other data capture vendors favor the latter. In practice, both may be used for different parts of the same data capture system.

In most installations, the data capture terminals will be located in the same building, or at least in close proximity to the computer acting as host. The discussions which follow assume that this is the case. If your situation is different, you will need to explore data communications options, but a full discussion of data communications is beyond the scope of this paper. The principles which will be discussed, however, are applicable to both remote and local data capture terminal networks.

For Hewlett-Packard data capture terminals, two options are available: multipoint connection, and point-to-point connection. Which of these options you choose should depend on:

- Transaction volume
- Volatility (i.e., minimum required reliability)
- Proximity to the computer site
- Cost
- Flexibility

The first of these, and the one usually recommended by Hewlett-Packard, is multipoint. Hewlett-Packard offers two different hardware implementations of this software protocol.

Standard multipoint requires that all data capture terminals be connected in a "string," i.e., "daisy-chained" together. Cabling this arrangement over a large factory may be somewhat awkward, and the arrangement tends to be inflexible. Adding a terminal generally requires extensive additional cabling and in most cases will require that the data capture system be shut down during the installation. In addition, the method is limited in the distance that it can cover, and its immunity to electrical interference is somewhat limited.

The immunity of ordinary multipoint to single-point failure is limited by the length of the string. If a terminal fails, it will usually prevent operation of all terminals beyond it in the string.

The Factory Data Link, or multidrop, is an arrangement wherein many terminals share a common "supply" line, rather similar to houses along a water main. Physically, the data link consists of a fairly thick cable which may be initially installed as a loop around the inside of a building. The data link is limited to a maximum of five miles in length or less depending on the locations of terminals relative to the computer site. Data capture terminals are "tapped" into this line in much the same way that sprinklers are "tapped" into a garden hose. Each "tapped" terminal may have more terminals daisy-chained onto it as ordinary multipoint terminals. The method is very flexible, since installing a new terminal involves only tapping into the line at the point closest to the site of the new terminal, or daisy-chaining off an existing terminal if a second terminal is to be added to an existing data capture site. Adding a

terminal disables only part of the system, and that only for a few minutes. Removing a terminal from the factory data link will usually not cause any downtime. Noise immunity characteristics of the factory data link are excellent; the data link will often function in environments with such high levels of radio frequency interference that no other connection method will permit error-free operation.

Immunity to terminal failure is very poor, however. Failure of a single data capture terminal can potentially "hang" all other terminals sharing the Factory Data Link.

Both multipoint connection methods, the daisy-chain and the Factory Data Link, are quite susceptible to single-point failure since many terminals share, at a minimum, a common computer interface and communication line. Failure of either of these components will cause the failure of all terminals in a network. For this reason, it is usually best to split a network into two or more "subnetworks" if a multipoint protocol is to be used.

If transaction volume is very high **and** the amounts of data transferred per transaction is large, performance can suffer. Because only one terminal can be using the common line at any given time, one very busy terminal can use most of the communication capability available. This must also be considered when designing the data capture network topology. But because of this facility sharing, the communication cost per terminal is quite low.

Point-to-point connection of data capture terminals has few of the disadvantages of multipoint connection. Immunity to terminal failure is excellent, since all common elements in the communication facility are thoroughly isolated from the terminal itself. Failure within the communication interface may easily be remedied by switching plugs **at the computer site**, an option not easily available with multipoint connection.

The chief disadvantages of point-to-point connection are the need to run separate cables for each terminal to be connected, resulting in decreased flexibility and significantly increased cost, and relatively low immunity to electrical noise.

Designing the User Interface

One of the most significant roadblocks in implementing a data capture system, and one of the least quantifiable, is simply **getting people to use the system**. One of the key considerations in this is the "user interface" - that part of the system which the user sees.

An almost universal characteristic of a data capture system is that the operators doing the most input usually have the least to gain from having the input done. This is in sharp contrast to batch systems or "on-line batch" systems in which the operators **primary** responsibility is to enter data. For the operator of the data capture system, then, maintaining the system is a burden rather than a part of the job.

By properly designing the user interface, this burden can be made as small as possible, and some element of reward can be introduced to make using the system more palatable. In general, the interface designer should be concerned with **simplicity** and **feedback.**

Simplicity Most data capture transactions will involve several steps, possibly implemented with a question-and-answer dialogue. If possible, this dialogue should be designed out of the user interface through the use of machine-readable documents. For example, consider the dialogue which might be required to move an assembly from one workstation to another. The important elements of the transaction might be the work order number on which the assembly is being built, the workcenter being moved from, the workcenter being moved to, and the quantity being moved. The system should generate all ancillary information such as the date and time of the move. In a dialogue, the transaction might be:

```
Computer:  W/O#?
User:        4158245 <enter>
Computer:  FROM OPN?
User:        45  <enter>
Computer:  TO OPN?
User:        50  <enter>
Computer:  QUANTITY?
User:        100 <enter>
```

This transaction requires 18 keystrokes and requires waiting for computer response after each of four entries. (These count a single initial keystroke for beginning the transaction.)

The transaction can be shortened somewhat by using defaults:

```
Computer:  W/O#?
User:        4158245 <enter>
Computer:  FROM OPN 45?
User:        <enter>
Computer:  TO OPN 50?
User:        <enter>
Computer:  QUANTITY 100?
User:        <enter>
```

This example requires only 12 keystrokes, since the application tries to anticipate the user's responses by checking existing information about the work order.

The transaction can be reduced still further by using a computer-generated work order form with bar codes pre-printed:

```
User:      (Wands composite FROM)
Computer:  W/O# 4158245: 45 TO -?
User:      (Wands composite TO)
Computer:  QUANTITY 100?
User:        <enter>
```

This example requires only one keystroke and two bar code reading operations. The "composite" is a bar code digit string containing the work order number, the operation number, and a transaction code meaning "move assembly." The operation code is ignored during input from the TO? query, e.g.:

041582450045

Thus, by careful choice of user interface, 18 keystrokes and four waiting periods have been reduced to three keystrokes or user actions, and only two waiting periods. These waiting periods can be made quite short with proper application program design.

Feedback Because the primary responsibility of the data capture operator is typically something besides data entry, it is important that he or she receive some kind of feedback or reward in return for entering the data. This will usually consist of an acknowledging beep or message, but should include some additional information when possible. This may be difficult to achieve, but might come in the form of a count of units of production, e.g., number of units tested today or total from this lot. This serves to give the operator a sense that something inside the terminal is listening. Giving the operator a printed summary of the previous day's input can serve the same purpose. In general, the important point will be to insure that the operator realizes that some concern exists that the data entry part of the job gets done as well.

The amount of feedback should be proportional to the amount of work required for keying in. If the operator is simply required to run a bar code wand over a tag on the assembly, or to drop a card into a slot, little feedback will be required. If some dialogue is required, as in the above examples, more feedback should be provided.

Designing the Application System

Designing application programs for a data capture system requires attention to some aspects of programming that are not normally given much weight in ordinary batch or on-line processing. In particular, the programmer for a data capture system must pay attention to:

- High transaction volume
- High reliability and fail-soft operation
- High volume reporting

High transaction volume

When the number of transactions to be processed is very high, special attention must be given to such aspects of application design as file locking, transaction selection methodology, and general response time optimization. Because the key to all a successful data capture operation is a good user interface, and because a good user interface will derail the operator's primary activity for as short a time as possible, every effort must be made to shorten response time. An execllent paper on this subject was presented at the North American meeting of the HPIUG in 1983[1]. This paper covered various aspects of application design relative to response time and overall performance.

Using IMAGE

Of considerable importance to maintaining adequate transaction volume is the use of record-level locking when using IMAGE. Because many operators are likely to be accessing a fairly small group of high-use records, attention should be given to minimizing locking conflicts; locking at the lowest possible level is therefore a necessity. In addition, PUTs to detail data sets should involve primarily data sets with few paths. Sorted chains, unless used for maintaining chronological sequences, should be avoided. In addition, if several records on a chain must be manipulated, as in the case of the work-in-process tracking example above, the records should be manipulated in core to minimize the number of physical I/O's required to finish the transaction. Programmers should be willing to sacrifice simplicity for performance in this response-time critical situation.

Menu Selection

Many traditional menu selection schemes rely on process handling, creating and launching a new process for each transaction, or group of similar transactions. The overhead of this operation is unacceptable in for high transaction volumes and should be avoided. Other methods, such as using the menu (function selection keys on the terminal) to selectively call procedures in a single large program file, will improve response time and decrease I/O and memory resource usage. Since program code is shared among all users of a particular program, considerable simplicity of design can be achieved along with a performance increase by using this method. Common data base access

routines should be placed in a single segment and shared among all transaction executors.

One helpful system design method is to avoid hard-coding menu parameters in your application program. If the program is designed to configure the function-to-key correspondence from a file, it becomes possible to provide each terminal in the data capture system with a subset of functions from a single large library. This feature becomes increasingly useful as your organization gains experience using the data capture system, since functions can easily be moved from terminal site to terminal site as the initial expectations inherent in the original design are modified.

Reliability

More than any other computer application, data capture systems require highly reliable software. While all software should be bug-free, data capture software with its limited communication back to its users, and its users' extremely limited computer training, is particularly dependent on non-stop, predictable, and accurate operation. The system should be designed to the greatest extent possible to be immune from problems from system failures, data base problems, program bugs, and especially user input.

Immunity from system failures can be provided in a number of ways. For a data capture system using IMAGE, the best way seems to be user logging. While requiring extra programming and extra design attention to be used effectively, transaction logging can provide up-to-the-minute backup for a busy data capture system.

Data base problems represent a broad class of difficulties such as broken chains in IMAGE data bases, DATA SET FULL errors and their equivalents in non-IMAGE systems, concurrency problems, and other difficulties in accessing or updating data on an otherwise healthy system. In general, the application should perform whatever checking is necessary to **commit the transaction to succeed.** This means checking data set capacities, insuring that records have not changed across locks, and perform whatever logical (data dependent) checks are necessary before actually updating the data base. Problems which are found should be reported to a central controlling process which can shut down the data capture system and inform the computer operator before any damage is done. In addition, the data capture system must **immediately** inform its operators that the system is no longer available. Otherwise, the operator may continue to "input" data, unaware that the system is no longer listening!

In order to prevent program bugs from causing serious damage, programs should be self-checking whenever possible. For example, if inventory counts are kept in several places (possibly split in several different ways), any time two or more counts are available, they should be compared. If an error is found, the problem should be reported to the operator, and depending on the potential seriousness of the problem, the data capture system should be shut down automatically. This strategy will prevent a "cancer" from occurring because of later transactions relying on the incorrect data.

In some cases, it is possible to repair damage to the data base in real time. For example, if it is possible to check a summary field against its details, this should be done whenever it is not inconsistent with performance. (This does not mean that the data check must execute rapidly; it may simply be performed as part of an infrequently-used function.) In the event of a discrepency, the damaged summary field should be repaired and the error logged. In this way, it is possible to insure maximum availability of the data capture system, while simultaneously maximizing its reliability.

High-volume reporting

In using a data capture system, a large amount of data can be recorded for later use. While it may seem obvious at the outset that these data will be too voluminous for repeated reporting, this fact should be kept in mind throughout the design of the data capture system. Reports should be provided with significant "filtering" capabilities so as to produce meaningful information from the huge amount of data collected.

User Training Program

Data capture systems present a special set of user training problems in that the system's users will typically have a lower level of computer education than a data entry operator. In order to mitigate this problem, the application should be designed so as to minimize the amount of training required. If a particular transaction requires more than one step to complete, the data capture system should guide the user through the transaction in such a way that only one path is possible. Error messages **must** be phrased in the user's "native" language; computer terminology should be avoided. **An uninterpretable error message will not be interpreted!** Failure to recognize this fact will mean lost transactions as users ignore rather than report cryptic error messages.

With this in mind, an effective user training program will be one which stresses training by example. For single-step transactions, the data entry becomes just one more step in a routine that the user is as already accustomed to as part of his or her primary duties. A multi-step transaction will require that the user understand something of the information that he or she is asked to present.

V. Evaluating your Data Capture System

After your system has been in operation for several months, you should begin to evaluate its performance and select changes to be made. Evaluating the system's performance relative to organizational objectives is beyond the scope of this paper, but four major operational areas should be evaluated:

- Reliability
- Operator acceptance
- Information availability
- User acceptance

Reliability

One repeatedly-stressed factor in a successful data capture system is its reliability. Uptime over the evaluation period should be measured, and any problems analyzed. Have there been communication failures? Have many problems been found by self-checking routines? Have the automatic repair and shutdown facilities been effective?

Operator acceptance

If the system has been reliable and available, it is important to measure operator resistance to the system. This is normally high at first, and then gradually decreases as data entry becomes a normal part of the operators' routine. (If the system has not been accessable, resistance will usually remain high regardless of other factors.) If resistance to use of the system is high after the system has been in place, the causes should again be analyzed. Have terminals been put in the right places? Are there enough terminals to prevent operators from standing in line? Is response time low enough? Is the user interface too complex, thus imposing a burden on the operator? Is insufficient feedback being provided?

Information availability

Since the primary purpose of a data capture system is to provide information which was too costly to gather with ordinary batch or semi-batch processing techniques, the first evaluation is whether the system is capturing the data it was designed to capture. If the operators have accepted the system, information should be available. If not, the design factors used relative to timely and useful presentation of information may need to be reexamined. Can reports be produced on an exception basis if desired by users? Can information on reports be tracked directly to physical activity monitored by the data capture system? Can information be obtained on-line so that users can "watch" the system work? Are users confident that they know the source of all of the data presented to them?

User acceptance

Finally, if information is available, properly presented, users should accept the system. Acceptance means using the results of the data capture system in their daily operations. If in fact decisions are made on the basis of information captured by your data capture system, it is by definition successful. After all, that is the end purpose of information systems!

Bruce Toback wears many hats at Infotek Systems, which keeps him very busy, indeed. What free time he does have is being currently spent in acquiring a pilot's license.

Rick Simon
Performance Specialist
Hewlett-Packard

MPE V Performance Results

It was the policy of the Program Committee to allow speakers on the conference Program if and only if their papers were submitted in time to appear in these proceedings. For the most part we adhered to that policy. However, our first concern was the quality of the Program and the timeliness of its topics. Those concerns took precedence over all others. Therefore, when the author of this paper indicated that, for a variety of reasons, he was unable to prepare a paper in advance we allowed him on the Program nevertheless.

The author will present the latest performance results derived from testing MPE. We of the Program Committee thought that it would be more useful for conference attendees to receive up-to-date information on such an important topic rather than older information printed in advance for the sake of completeness.

Signed:

IUG ANAHEIM
PROGRAM COMMITTEE

Structured Tuning: A Layered Approach to Managing System Performance

Steve Wilk & Sam Boles

Hewlett-Packard

Synopsis

You've seen the benefits of Structurism: Structured Design, Structured Programming, Structured Implementation have enabled new achievements in the quality and timeliness of computer systems. Here you'll see the Top-Down principle extended to the perennial subject of System Performance in the form of Structured Tuning.

A team of performance engineers from Hewlett-Packard have developed the Structured Tuning Methodology that starts at the top-level Network and System Layers with the most global factors of system performance -- CPU, main memory and I/O -- and iterates down thru intermediate Layers including Subsystems and Applications dealing with system tables and database control blocks to the low-level Process and Procedure Layers sensitive to individual file placement by logical device and individual algorithms.

You'll see Structured Tuning use a variety of performance tools -- supported, contributed, consultant and proprietary -- to interface with the comprehensive instrumentation integrated with the MPE software, to help you identify performance bottlenecks. You'll see how Structured Tuning protects against Premature Optimization and enables you to focus on high-yield performance efforts. And you'll see how Structured Tuning complements its reactive remedial tools with proactive modeling techniques to enable the Ounce of Prevention Mode.

Background: Techniques, Topology, Tools

The Right Problem

Once in another time and another place, there was a man who had two problems. One of his problems was that he wanted to play a tune on his fiddle. He thought if he solved one of his problems, then he'd have one less problem. So he played a tune on his fiddle, and, sure enough, he got *that* problem solved.

However, in the meantime, *Rome burned.*

He'd solved a problem -- but it was the *wrong problem.*

The Other Cost

We all know there's no such thing as a *free lunch.* Maybe there's no such thing as a *free anything.* To do anything has a *cost.* In fact, it has *two costs.* One cost is what it costs you to do it. The other cost is what it costs you not to do something else. You call that the *opportunity cost.*

A lot of the time it's the opportunity cost that really hurts. You've got a limited budget and limited resources. If you use your limited (and expensive) performance engineering resources to solve the *wrong problem,* the opportunity cost may be that your *real problem* doesn't go away.

The Techniques: A Method to Your Madness

Structured Tuning is a methodology that helps you identify the *right problem* to work on. Structured Tuning decomposes the complexity of a computer system into a series of hierarchical layers that can be traversed from the top down.

Premature Optimization

This top-down discipline works against *Premature Optimization,* one of the costliest parts of the Wrong Problem Syndrome in performance tuning. It forces you to look at the *global issues first,* before getting down to the local issues. If you start at the top and look at the layers in order, you see things in the context of the "Big Picture." This helps you identify the high-yield projects and get your *priorities straight.*

Like Structured Implementation . . .

Structured Tuning does for performance what Structured Implementation does for a project: You identify the main line and build a prototype, fast and cheap. You try it. If it doesn't work because the technology can't stand the strain of the application,

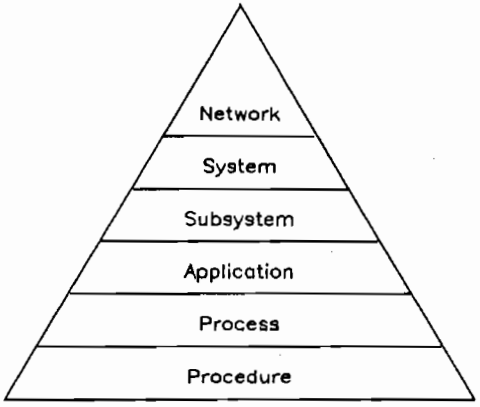
you throw it away. If it's not what the user needs, you throw it away. You use the prototype to solve the *right problem*: *Can it be done? Should it be done?* . . . and if you get a negative on either question, you throw away less code and minimize your losses. If the prototype registers positive, you get into the inevitable rewrite faster and on firmer ground. The Structurism of Structured Implementation helps you ask the *first questions first*; and the same thing happens in Structured Tuning.

Stepwise Traversal

Stepping down thru the hierarchy, if you see at the System Layer that you're bottlenecked on disc I/O, you don't jump right in and rewrite that simulation algorithm (that you *know* is killing you) to cut its CPU consumption in half. That might not be a bad idea . . . eventually . . . but not now. This discipline has Pareto's 80-20 Rule built into it, and gives you a first-order prioritization of your tasks.

The Topology

A dissection of the subject that seems to yield a workable structure for Structured Tuning has the Network as its top Layer and decomposes down thru System and Program Layers to the Procedure Layer:



The Structure of Structured Tuning

Metrics

The "bottom line" is *productivity*. The metrics we use are transaction throughput ("how many orders can I enter in a hour?") and response time ("when I key in 'how much is 2 + 2' how soon do I get the answer after I touch RETURN?")

Tools

To get these metrics in this range of layers there's a range of tools. Some are software products you can buy. Some are contributed. Some are part of your HP Performance Specialist's bag of tricks. These tools for the most part interact directly or indirectly with the comprehensive performance measurement instrumentation that is part of MPE.

A key measurement tool in the HP3000 domain is OPT (On-line Performance Tool). This tool is a supported product and provides a variable-interval, variable-level window into the system.

Complementing OPT is a consulting tool MPEDCP (MPE Data Collection Program) which collects performance statistics similar to those in OPT, but with a somewhat different focus and format.

For finer granularity at the file/process level, IODCP is a consulting tool that gives statistics to help with mid-range and low-level tuning.

At the Process and Procedure Layer the APS (Application Program Sampler) system provides insight into CPU utilization down to the level of lines of code.

In addition to these tools, there are MPE and subsystem commands and utilities that enable views of how the system's running, eg SHOWCACHE, DSCONTROL, DSDUMP.

Modeling

Complementing the performance monitoring tools are the models of the "real world" that can give valuable support to Capacity and Configuration Planning and Performance Management: Benchmarking, Simulation, Analytic Modeling.

The Ounce of Prevention Mode

While the monitoring tools tend to be reactive to problems that already exist and need to be corrected, models are more predictive and proactive. With predictive power, models can provide the information necessary to make a good decision at the beginning that can prevent a problem rather than fixing it after it's happened. This *Ounce of Prevention Mode* can save money, time and human resources.

Benchmarking

In Benchmarking, the first degree of abstraction from the "real world," we model the range of system and workload configurations that is proposed or perceived to be a correct representation of the current situation and/or the future. Being a model only a step away from the real thing makes Benchmarking tend to be very costly in time and other scarce resources.

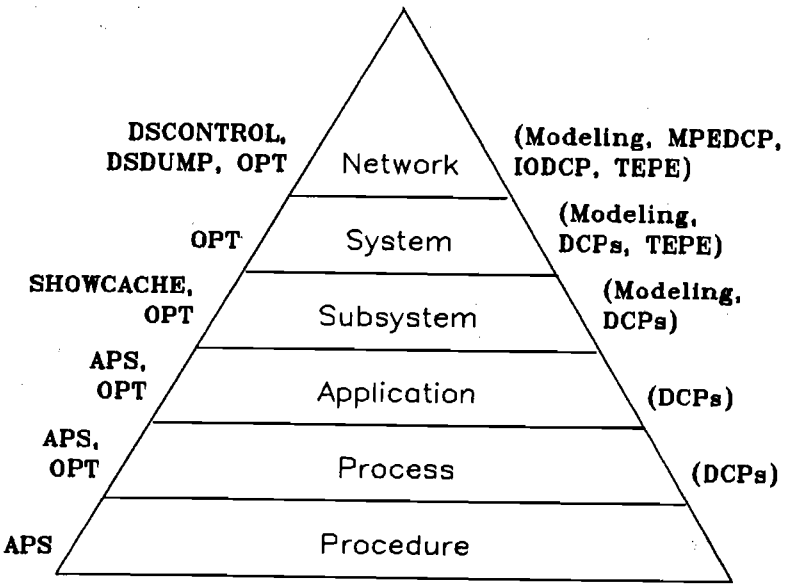
For benchmarking, TEPE (Terminal Emulator and Performance Evaluator) provides a direct hardwire connect from the ADCC on the driver system (Series 4x) to the ADCC or ATP on the System Under Test (Series 4x or 8x). Guided by terminal "scripts" TEPE sends bit-serial messages that look to the HP3000 under test like inputs from CRT's direct-connected to it.

Simulation

The second degree of abstraction from the "real world," Simulation, models a trace of the "real world"; for example, a detailed sample of disc I/O traffic, taken from a production system is modeled to predict its behavior in a proposed environment, eg a system where disc domains are cached in main memory.

Analytic Model

The third degree of abstraction from the "real world," the Analytic Model, models algorithmically the functions of the "real world"; for example, a queueing-theory based central server network that includes a CPU of variable speed and a variable number of discs with variable speeds and variable user classes and variable think times and variable ... and variable ... This model is typically the fastest and cheapest, can be the most flexible, is sometimes the only feasible, but often is the most difficult in some ways since it requires us to *understand our data and programs and systems in great depth.*



The Mapping of Tools in Structured Tuning

System Components

CPU

The Central Processing Unit is a key variable in the Structured Tuning Methodology, but in the typical commercial job mix it's usually not where you get the primary performance bottlenecks. In the current HP3000 line you have a .4 MIPS processor in the 4X series and a 1.0 MIPS processor in the 6X series for vertical flexibility. For horizontal flexibility, the HP DSN (Distributed Systems Network) provides a variety of 3000-to-3000, 1000-to-3000, and 3000-to-other linkage possibilities.

CPU and Disc Caching

Any performance tuning methodology today in the mid-range to high-end of the HP3000 must examine the implications of Disc Caching. This may be the most significant performance contribution in the HP3000 family since the 1 MIPS processor; it makes something beyond a brute-force raw-horsepower performance improvement. Disc Caching needs 20-40% of the CPU in order to give you really substantial (2X, 3X) performance gains.

If the necessary CPU is not available, Disc Caching can degrade performance.

Main Memory

In the HP3000 family you've got this range of main memory available:

Series	Min	Max	Write Speed
68	2Mb	8Mb	375 ns
48	1Mb	4Mb	530 ns
42	.5Mb	3Mb	530 ns
39	.5Mb	3Mb	530 ns

On the high-end, in the 6X series, you have an 8K byte high-speed (ECL RAM) memory cache that has a 95% hit rate. These gives you an effective 145 nsec memory read speed.

Main Memory and Disc Caching

Let's put main memory in the Disc Cache perspective: The essence of Disc Caching is to let you get at your disc data at main memory speeds rather than disc access speeds. If you compare the two, you find that accessing main memory is something like 10000X to 50000X faster than your average disc memory. Since disc I/O makes up such a large part of commercial data processing, that kind of data access improvement can really pay off in better response time and throughput rates. If your job mix has a 70% cache hit rate (as is often the case in commercial applications, you can pay the 3 or 4 msec price for Caching, save 30 or so msec on 70% of your I/O, and get a good return on your investment. To give Disc Caching room to work, you need to give it 1 or 2 Mbytes of main memory on the high-end machines to get the 2X and 3X performance gains it's capable of producing.

If Disc Caching is enabled on a system already under memory pressure, it may degrade performance.

Disc Memory

Disc I/O has been the perennial performance bottleneck in the typical HP3000 job mix -- or *so it seems*. With the arrival of the 4X and 6X machines, the old 30 I/O's per second max has gone away. With multiple master drives (each with its own controller), on multiple GIC's (General I/O Controller) (each with a 1 Mbyte bandwidth), on multiple IMB's (InterModule Bus) (each with a 3 Mbyte bandwidth), we can sustain 60 to 120 or more disc I/O's per second.

But we don't.

Why?

Because *we don't ask for it.*

What we usually do is port our jobs 1-for-1 from the Series III to a 44 or a 64, configure in 2 or 3 times as many terminals as we had before to go against that single IMAGE database that we built 4 years ago when we were 1/4 the size we are today. And we sit there, queued up on a single-threaded Database Control Block with the CPU only 50% Busy and a disc subsystem capable of 100+ I/O's a second being asked to do 50 I/O's a second.

Disc Memory, Disc Caching and PEP

For the high-end 3000 installation, one of the important disc drive innovations in recent years is the 400 Mbyte HP7935.

When we first heard about the 7935 we were really impressed: for a small percent more than you pay for a 7925 you could get more than 300% of the 7925 capacity. Then we ran some benchmarks and found that for the capacity gain you paid a 15-20% performance penalty.

Then came PEP (Performance Enhancement Project.) The engineers in Boise tweaked better than 15% improvement into the 7935 and got it within noise-level of the 7925 performance.

The PEP microcode structure gain coupled with Disc Caching, Buffer Prefill and RPS (Rotational Position Sensing) makes the on-line storage economy of our 400 Mbyte disc drives (7933, 7935) a viable solution

for the performance-sensitive customer whose "results must be measured by performance."

But remember: Disc Caching needs file locality, a good (3+:1) Read:Write ratio and a heavy disc I/O load to produce the 2X and 3X performance gain it's capable of.

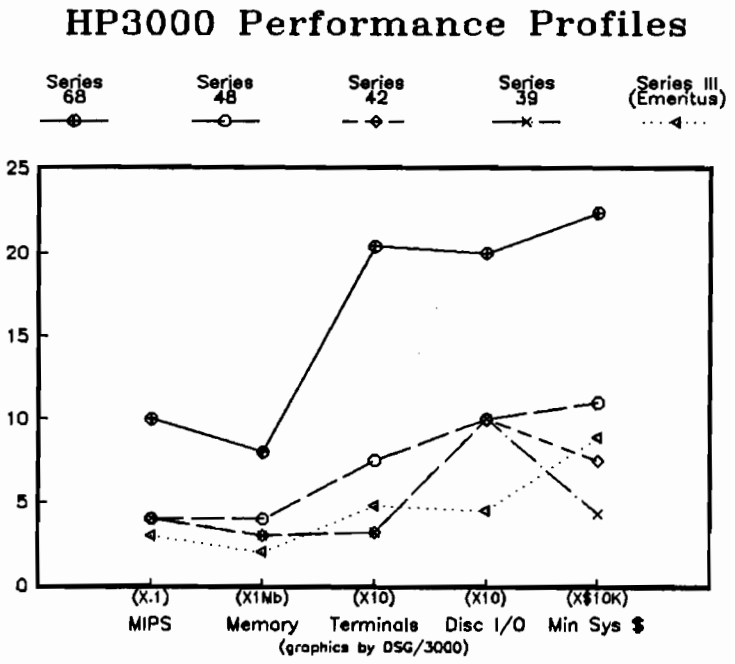
Other System Components

In addition to the Big Three -- CPU, main memory, disc I/O -- your system performance can be influenced by factors like:

- Datacomm line speeds for terminals and distributed systems
- Terminal/Workstation speed and intelligence
- Mag tape speed/density (especially in back-ups)
- Printer capacity
- System tables

Structured Tuning looks at these components as they emerge at the various Layers of the hierarchy and assigns them a direction and magnitude appropriate to their station in life. This is usually very application- or installation-dependent.

Here's an "artist's conception" of the performance characteristics of the HP3000 family:



Structured Tuning in Action

Now you've seen the topology of Structured Tuning, the objects that it deals with and some of the tools that it uses. Let's take a look at some examples of Structured Tuning in action.

The Network Layer

In some ways, this one may be the toughest. You've got a wide range of configurations, line speeds, ranges, types, sharing, simultaneity, concurrency, costs, metrics, vendor compatibility issues.

From the aspect of Structured Tuning, there are some rules of thumb that seem to work. Aim at doing fewer things bigger. From the performance aspect, one user with a large record is better than multiple users with several small records. One thing you can check for is a small overrun on your record size; for example, if you configure 1024, but send 1026 you may generate a lot of continuation records.

If at the System Layer you find a CPU bottleneck that traces to the Application Layer where you find that the heavy crunch of business graphics is consuming your CPU at the expense of general response time, consider relieving the CPU pressure by distributing the load. For example, an RS232 link to an HP9000 could enable the offloading of the heavy-duty processing to a high-performance, high resolution work-station solution. The artwork database in a convenient intermediate form could then be uploaded to the HP3000 for integration with text and output to the laser printer.

Notice the Structured Tuning Methodology at work here: it iterates thru lower Layers for details characterizing problems detected at a higher Layer, and may percolate up to the highest Layer for the solution.

The System Layer

A good starting point at this Layer is to look into the system thru the window that is OPT (On-line Performance Tool). At the global level, OPT shows you CPU, main memory and Disc I/O utilization on a variable interval basis, and CPU and Disc I/O utilization on a cumulative (multi-interval)basis.

Here's an example of the OPT global report (clipped on the right for printing convenience):

```
      RESOURCE USAGE DISPLAY      HP32238X.VV.16  OPT/3000
(C) HEWLETT-PACKARD COMPANY 1979, 1980                    SEBDC0L2 S64 2MB CACHE ON
   CURRENT INTERVAL:    66.5 SECONDS          OVERALL INTERVAL:    11.1 MINUTES
ACTIVITY IN CURRENT INTERVAL
                          10        20        30        40        50        60
      MEMORY USAGE M----MC-----------CS----SD--DK-----------------------------------

         CPU STATE B-------------------------------------------------------BP----------

                          10        20        30        40        50        60
 DISC I/O ACTIVITY SSK-------------------------------------K
-------------------------------------------------------------------------------------
ACTIVITY OVER ALL INTERVALS
                          10        20        30        40        50        60
         CPU STATE B-----------------------------------------------------------BP-------

                          10        20        30        40        50        60
 DISC I/O ACTIVITY SK-----------------------------------------K

MEMORY USAGE LEGEND:
  M  Resident MPE
  C  Code segments
  S  Stack segments
  D  Data segments
  K  Cached disc domains

CPU STATE LEGEND:
  B  Busy on processes
  P  Paused for user and/or memory management disc I/O
  I  Idle
```

```
  G  Garbage collection
  O  Memory allocation and ICS overhead

DISC I/O ACTIVITY LEGEND:
  U  User disc I/O
  S  Segment management  I/O
  K  Cached I/O
```

Here we may see excess CPU, main memory, disc I/O capacity, Pause for I/O.

If we have a disc I/O intensive load, we can drop to the next Layer in our Structure, the Subsystem Layer, and take a look at the disc subsystem.

The Subsystem Layer

Here we can collect a trace of the disc I/O traffic and simulate what its behavior would be with Disc Caching.

We can look at the read:write ratios. We can see if the traffic is sequential or direct. We can look at file access locality in time and space. And we can run the trace thru the Disc Caching Simulator to see what kind of hit ratios we'd get if we enabled Caching and gave it a Mbyte or so of memory to work with.

If it looks promising, we can put in a Meg of Main, enable Disc Caching and take a look at what it does for performance:

```
:SHOWCACHE

DISC    CACHE     READ WRITE        PROCESS                 % OF    CACHE
LDEV  REQUESTS    HIT%  HIT%  READ%  STOPS     K-BYTES     MEMORY  DOMAINS
-------------------------------------------------------------------------

1       16436       70    93    73   3776        1434         23      252
2       17809       69    95    66   3673        2319         38      358

Total   34245       70    94    69   7449        3753         62      610

Data overhead is 158K bytes.
Sequential fetch quantum is 96 sectors.
Random fetch quantum is 32 sectors.
```

We see we get fewer process stops (the Cache Manager does not interrupt on a cache hit for a read), get bigger and more level physical I/O's, put to use the idle CPU and main memory and, by happy coincidence, double the throughput, cutting the wall time about in half.

The Application Layer

For the next case study of Structured Tuning in action, let's look at what made it feasible to print multiple copies of the paper you're reading on an HP2680 Laser Printer. Let's say we have a technical publication that we've typeset on the laser printer, using 3 forms and 12 fonts and we've decided to print 120 copies of it on the system. We did a TDP (Text and Document Processor) "Final to *2680" to get TDP to format it, deferring the SPOOL file. Then we did a "HEADOFF 14" (to save a tree), set OUTFENCE up so we could get the right fold on the document and not get alien output commingled with our 120 copies. We then did an "ALTSPOOLFILE #Onnn;COPIES=120" and set the priority over the fence.

It's a 10-pager, so we go away and decide we'll come back in (10 x 120)/45) minutes. (We've got 120 copies of a 10-page document running on a 45 page-per-minute laser printer). We come back in about a half hour, smile pleasantly as we walk through the scowling crowd waiting impatiently around their/our laser printer. We reach for our 120 copies and find we have only 30.

We pull out our Structured Tuning notes, at the System Layer we run OPT, get the global display, find heavy disc I/O. We go into the Process display, see that the SPOOLER is doing a lot of I/O. We run SPOOK, find 1100 lines of dense environment file at the beginning of our output.

We then observe the laser printer. At the end of each document, as our heavy-duty environment file is downloaded, because of the size of the file, the laser printer infra-red fuser does a time-out, then goes into a warm-up cycle. This cuts our printer performance to 10 pages a minute.

We smile sheepishly at the irate by-standers, as we set our SPOOL file priority to 1, turn the Headers back on and reset the OUTFENCE to normal.

We then create a STREAM file to:

run SPOOK
text in our SPOOL file
APPEND the environment file once
APPEND 10 copies of the text
to the environment file
APPEND END to get a new SPOOL file

We run this in background, so no one gets impacted in response time.

When we get the new 10X SPOOL file, we go thru the HEADOFF etc routine we did before, set copies=12, and the printer runs at about 38 pages a minute rather than 10.

Structured Tuning at the Application Layer helped us isolate the problem. Structured Implementation built this *ad hoc* prototype to see if the technology could do the job. (The next step in the Structured Implementation Methodology is expected to make the job operationally feasible and give a 2X to 3X overall performance improvement.)

The Process Layer

Let's say that in our Top-Down Traversal of the Structured Tuning Structure, we establish at the System Layer that we have a performance problem that we trace thru the Subsystem Layer to the Application Layer. And let's say that the application has multiple users running the same program, and we think it's one of the processes of that program that's causing our problem.

We can get into OPT and go to the process display to try to further isolate the problem.

The Procedure Layer

If it turns out that our top-priority performance problem is traceable down thru the hierarchy to a given process, we may want to look at CPU consumption at the procedure level or even at the granularity of lines of code. APS (Application Program Sampler) can help us here and enable us to pinpoint the sector where the heavy resource consumption is happening. With this focus and granularity, we can have a good calibration on where we can best utilize our limited performance tuning resources.

In the APS CPU Utilization report below, we see that the GEN'CHAR procedure (which does a 2-dimensional raster scan of a character "ROM" within a 2-dimensional scan of a message list) is probably a good place to tune among the 10 procedures in the program -- 8 of which didn't have enough activity to register on the scale.

Conclusion

You've seen how the Structured Tuning Methodology is an iterative process that starts at the topmost global layer. It can help you keep a balanced perspective as you look at performance problems. You've seen some of the tools and how they're applied at the various layers to zero in on bottlenecks. And you've seen how Structured Tuning can help you find and fix the *right problem.*

Program Name: SEBPROG.BOLES.CAD - Segment: %301 SEG'
Distribution of Direct CPU Time Over Procedures
(95 Segment Samples = 5.26% Direct of Program Time)

```
+-------------|---------|---------|---------|---------|---------|+---%---%CUM--
SET'UP'CHAR'ROMDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD   | 45.3  45.3|
|   GEN'CHAR DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD| 54.7  100.|
+-------------|---------|---------|---------|---------|---------|+---%---%CUM--
Minimum bar threshold is .0%
```

APS CPU Utilization Report

About the Authors ....

Steve Wilk *is the Performance Center Manager at the Hewlett-Packard computer facility in Cupertino, California, with responsibility for performance benchmarks and product characterization. His 18 years in the industry cover a wide range, including programming, systems analysis and program management. With an undergraduate degree in mathematics, Steve received his MBA from the University of San Francisco.*

Sam Boles *is a Systems Engineer in the Systems Performance Center at the Hewlett-Packard computer facility in Cupertino, California. With HP for seven years, Sam's computer experience started back in the AUTOCODER days of the 1401/1410, migrated thru the 360/370 era, and now focuses on HP hardware and software performance characterization. Sam earned his MS at UCLA in Information Systems.*

sebiug37 2345 14Nov83

OPTIMIZING SYSTEM USAGE IN A MULTI-CPU ENVIRONMENT

Joel Martin
Project Resources, Inc.

OVERVIEW

As a customer's need for extra data processing power increases, a common, and often appropriate, response is to purchase additional HP3000 CPUs or to upgrade to larger HP3000s. When faced with this prospect and when the current environment already includes two or more CPUs, very substantial cost savings may be realized by a careful review of the existing hardware, application software and other elements of the total picture. This case study summarizes the approach used and results obtained by one large company with eight HP3000 CPUs.

BACKGROUND

In this case the HP environment consisted of eight Series III CPUs, 30 disc drives, 11 tape drives, 10 line printers and 384 ports, all purchased within a three year period. There was one additional CPU (with peripherals) on order and there were open requests for two more. Many of the ports were shared by users through the company's port multiplexor, so the number of terminals seemed to be staggering.

This hardware was spread unevenly among three autonomous departments. There was a clear need for additional horsepower in each department, and each department's solution called for additional hardware purchases. Each department had its own operations group ranging in size from two to nine people.

At the division manager's level there were a number of questions:

1. Were the computers being used to their fullest potential?
2. Were the current computer resources distributed over the three departments in a roughly equivalent manner?
3. Did planned growth warrant one, let alone three, additional CPUs? and
4. How could efficiency be improved and costs cut?

A team of three senior people, one from each of the systems groups, was assigned to review system usage and develop a methodology for addressing the division manager's concerns.

ASSESSING THE EXISTING SITUATION

HARDWARE INVENTORY

It was immediately apparent that the first step should be to introduce the three team members to each other. (The groups were quite autonomous!) The next step was to inventory the hardware, exclusive of terminals. (Terminals were not included because the project was to focus on the higher priced items and on service to the user. There was certainly a sufficient number of terminals.) Though each group kept its own inventory records, a manual

inventory was quickly done by visiting each of the four computer rooms and counting CPUs, memory boards, disc drives and the other peripherals. A chart depicting all the CPUs and their configurations (memory, disc, tape drives and line printers) was drawn up.

SOFTWARE INVENTORY

To get a clear picture of the application software being used, personnel in each of the major functional areas (marketing, engineering, manufacturing, shipping, etc.) were interviewed. When the software inventory was completed there were found to be no fewer than 37 major applications being run on the eight CPUs and a host of minor ones. (How often we overlook word processing.) One CPU alone was found to be running eight applications.

A chart similar to the hardware inventory one was developed and relative usage weightings (light, moderate, heavy) were assigned to the applications as well as a designation as to whether the application was overwhelmingly interactive (nearly no evening batch processing), overwhelmingly batch (heavy evening and cyclical usage) or simply overwhelming whelming (heavy day usage and heavy evening reporting). Functional relationships between applications were noted and peak periods were identified. Data relationships within and between applications were also identified as well as any tape or DSLINE data exchanges.

STAFFING

One department had two operators for a single machine (to insure coverage in case of illness or vacation), a second department had eight operations personnel for four machines (heavy batch production and two computer rooms in different buildings), and the third department had one full-time and two half-time operations personnel for its three CPUs. The ratios of operations personnel to CPUs were 2:1, 2:1 and 2:3. The number of operators was found to be primarily constrained by application system design.

SYSTEM USAGE

Line printer usage was roughly measured in terms of boxes of paper printed per month per line printer.

Disc space usage and disc leveling on all machines was checked using FREE2. Data base capacities for all major applications were compared against high water marks to determine if there was hidden waste of disc space.

Disc I/O, memory and CPU utilization, system table sizes and program performance were analyzed using OPT/3000. Basic computer usage information available through the REPORT command was collected regularly by operations personnel over a two month period.

Determining whether the throughput was acceptable proved to be less easy to define. Three barometers of processing sufficiency were defined:

1. Was the computer processing all the work required in the time available? (Particularly batch.)

2. Were the people whose jobs required heavy computer usage having to work overtime due to the 3000s inability to process peak loads?

3. What was the level of user frustration, if any?

Throughout the measurement period, the focus remained on the systems as a whole rather than the efficiency or performance of particular programs running on the systems.

ASSESSING THE SITUATION, COMING ATTRACTIONS

The rapid fire purchase of HP3000 hardware had been necessitated by the bringing on-line of new applications systems, the cutover of enhancements to existing systems and the increased workload due to sales, manufacturing and shipment increases related to order activity and product introductions. A best effort to forecast additional machine requirements meant plotting the expected increases in sales, manufacturing and shipping against resource requirements. (E.g., if sales were expected to grow 50% in the next year, then the volume of order processing computer activity could be expected to grow 50%. If order processing required a fully configured Series III, then upgrading to a Series 44 would be sufficient for only one additional year. So either the 44 should be leapfrogged and a 64 considered or the application should be split and distributed over two or more machines.) The resource requirements for planned and proposed systems were mapped according to the same criteria as existing systems (batch or interactive? cyclical? light, moderate or heavy usage?) and the impact of planned enhancements to current systems was forecast as well. (What would IMAGE logging do to order processing requirements? How about the planned addition of sales order audit trails to the order processing system?)

The forecasting effort did not lend itself to the simplistic numerical analysis of OPT/3000 so appropriate percentages were guessestimated and relative terms (e.g., planned enhancement 'A' would be roughly equivalent to an additional MRP run or would add 10% more overhead) were employed.

The forecast also took into account what was known or guessed to be forthcoming from Hewlett-Packard. Known items, e.g. an upcoming release of MPE promising performance improvements, were included in the plan as well as heavily rumored product announcements. (When this study was undertaken it was widely rumored that a substantially more powerful CPU than the 44 was on its way. The announcement of the 64 followed by one month the decision to cancel the open order for the 44 and two pending orders for 44s.) Rumors obviously carried less weight in purchase planning than known facts and were discounted entirely unless substantiated to some degree by trade publications or industry sources.

ISSUES AND ANSWERS

By this time the overall picture had developed quite clearly. The methods of optimizing company wide system usage were determined to be:

- o Shift Applications Among CPUs.

  The objectives here were to

  1. Establish better interactive vs batch processing balances to reduce contention, especially at peak periods,
  2. Place functionally related applications on the same processor to increase efficiency and control, and
  3. Spread applications with similar processing cycles across different CPUs to avoid bottlenecks.

- o Reduce Functional and Informational Redundancy.

  1. Separate databases for each manufacturing line were to be consolidated (when on the same CPU),
  2. Manufacturing costs and similar calculations being done in two or more areas were to be performed in only one area, and
  3. Vendor files, customer files and the like were to be broken out into stand alone databases to be shared by many applications rather than each application carrying its own version.

- o Monitor System Performance.

  1. A plan was drawn up to regularly monitor system performance via CPU and connect time tracking at the system level and expanded use of OPT/3000,
  2. Additional training for operations on the use of diagnostic tools was scheduled, and
  3. The development of simple application specific performance analysis tools was planned.

- o Consolidate Operations.

  1. The three operations staffs were consolidated into a single operations group,
  2. The operators were cross-trained on systems in other departments, and
  3. Batch jobs were run concurrently as much as possible instead of sequentially. (This alone accounted for a 30% throughput improvement).

- o Establish Guidelines

  1. Guidelines for the development of efficient IMAGE and KSAM based systems were developed using performance tips from IUG articles as a starting point,
  2. A plan was drafted to keep development people up to date on software changes and users group articles, and
  3. The group elected to take advantage of HPs performance consulting when it seemed appropriate. Members of the development staff in the affected areas were always involved with HPs analysis.

- o Level Hardware

  Memory and disc drive resources were spread more appropri- ately

among the eight CPUs based on the application weighting and OPT/3000 results from earlier phases of the project.

- Define Adequate Levels of Support

  Meetings were held with the key users to review system per- formance on a regular basis and forestall response time pro- blems.

- Eliminate Non-essential Processing

  There goes the word processing. Standalone word processors were purchased to replace the systems available on the 3000.

RESULTS

The purchase order for a new 44 was cancelled and the two proposals for additional CPUs were rejected.

The division came to view the HP3000s as shared resources and started to realize economies of scale.

Proposed changes in the operational procedures were adopted and up to 30% throughput improvements were realized.

As the company's sales continued to grow, so did the need for computer resources, but at a markedly slower rate.

CONCLUSION

Optimizing performance across several CPUs requires flexibility in usage and allocation of human and hardware resources. If the analysts, programmers, operations staff and users function as part of the same team, then overall improvement in system performance can be very drama- tic.

Mr. Joel Martin, Research and Development Manager of Project Resources

Joel Martin is responsible for the Research, Design/Development, Pro- duct Documentation and Product Testing functions of Project Resources.

Mr. Martin currently manages a team of HP3000 data processing professionals in the development of enhancements to PRI products.

Mr. Martin also acts as a Consultant and Guest Lecturer.

Prior to joining Project Resources, Mr. Martin spent four years as a Senior Programmer, Senior Systems Analyst and Programming Supervisor for ROLM Corporation. He was the lead Analyst in the design, development and implementation of their Order Processing Systems as well as a number of smaller marketing systems on the HP3000. During this time Mr. Martin did numerous Performance Studies of multi- and single-CPU environments. He was the sole delegate from a group of 60 Analysts to attend an International Users Group Meeting for the HP3000 in San Antonio, Texas.

Mr. Martin spent two years as a Programmer/Analyst with ASK Computer Systems where he worked on both the HP1000 and HP3000, developing the Physical Inventory Module and the Purchase Order Printing System.

Mr. Martin was part of IBM's Summer Technical Program in Poughkeepsie, New York where he participated in a PL/M compiler development project.

Mr. Martin has a B.S. Mathematical Sciences from Stanford University.

HEWLETT PACKARD

Features And Performance of HP<->HP

Data Communication Products

1984 HP International User's Group

Anaheim, California

Peter Hansen
Systems Specialist
HP-Fullerton SEO Technical Center

HEWLETT PACKARD

Introduction

Hewlett Packard's 3000 computer family has a number of HP-to-HP communication products falling into two categoies: cpu-to-cpu and terminal-to-cpu. 1983 and 1984 will see major changes in both categories of products. The biggest changes in 1983 included:

(1) The maturing and acceptance of X.25 technology:
   (a) Certification of many packet networks.
   (b) Certification of a low cost packet processor for use in small private networks, and also for use as a gateway to public networks (if wanted).
   (c) The introduction of a packet assembler/disassembler (PAD), or X.25 statistical multiplexor, by HP Grenoble. The HP2334 can connect 4, 8, 12 or 16 ports to a packet network at speeds up to 19.2 KB.
   (d) Terminal block mode support for VPLUS applications when used with the 2622, 2624 or 2627 CRT's.
   (e) Allowing a character printer on a 2334 PAD to be a shared printer. All X.25 HP3000 CPU's can build a switched circuit to a 2334 PAD connected printer. The FOPEN intrinsic will allow REMOTE NODEID's from the NETCON database to be used as a CLASSNAME.

(2) The Multidrop Terminal Support (MTS/3000) product matured by:
   (a) Adding support for the HP2333 cluster controller which permits all HP CRT's and character printers to be supported as if they were hardwired to an ATP.
   (b) Adding support for several HP line printers. This is especially useful when wanting to place a medium speed printer at a few hundred feet from the cpu, or across the country.
   (c) Adding both diagnostic and statistical reporting facilities to the product.

(3) Adding block mode terminal support to the Series 100 and 200 personal computers, as well as convenient file transfer programs from PC-to-PC and PC-to-HP Computer.

(4) The introduction of HPTELEX which enables an HP3000 to send or receive messages from the world wide TWX/TELEX network.

(5) The addition of autodial capability to dial-up DSN/DS-3000, MRJE/3000 and RJE/3000.

(6) The introduction of the Workstation Configurator product on the HP-3000, which allows you to custom modify the terminal ADCC and ATP driver tables to control such items as: hand- shake method, timeouts, timing delays, and input editing.

(7) Significant field organization changes: to better support datacomm; to better train HP field employees; and to better train our customers.

(8) Terminal voice output and terminal bar code input was introduced.

HEWLETT PACKARD

(9) Support of PBX's from Rolm, Northern Telecom and Intecom.

(10) Standardization of HP's Telesupport program for better CE and PICS response to your hardware/software interruptions.

(11) Regionalized PICS with full time SE's to provide long term continuity, and a better 'first call' success rate in handling your questions to your satisfaction.

As we look forward to 1984 we are anticipating making some further HP-to-HP communications enhancements:

(1) The DS/3000 product will be restructured so the X.25 capability is separated from the point-to-point product. Both resulting products will carry lower individual pricing.

(2) A base-band 10 megabit per second IEE-802 local area network DS product will be introduced:

  (a) Cable and cable interfacing hardware products.
  (b) Computer to network controllers.
  (c) Restructured software that is closely aligned to the International Standards Organization (ISO) Open System Reference Model (OSRM) seven layer architecture.
  (d) Network Management software.
  (e) Store and forward capability .

(3) New Series 100 personal computers will be introduced with both character and block mode support.

(4) A one-megabit local area network product for the Series 100 and Series 200 computers will be introduced.

(5) VPLUS X.25 block mode terminal support will be extended to more terminal models.

(6) VPLUS will be enhanced to support touch screens on one or more workstation products.

(7) Terminal-to-cpu speeds will increase to 19.2 KB for selected terminals when connected to the ATP.

(8) Satellite circuits will operate with greater efficiency than at present for cpu-to-cpu DSN/DS usage.

(9) Additional printers will be supported by MTS-3000.

(10) Software differences between ATP, ADCC and MTS connected terminals will be greatly reduced.

(11) A lower cost HP3000 will be introduced.

HEWLETT PACKARD

(12) A workstation multiplexor for IEE-802 LAN's will be introduced.

(13) An SDLC IBM batch print and job input product will be introduced.

As I look at what has happened, and what will be happening, it becomes obvious to me that the datacomm strengths of the HP3000 have become second to none in the marketplace. It gives me great personal pleasure to be an area level specialist devoting full time to helping you install and use HP datacomm products.

Let us look in some detail at the topics we have briefly introduced.

HEWLETT PACKARD

HP's DSN/DS Networking Capabilities

* Terminal Pass-Thru From One Host To Another
* Remote File Access By A Program At One Node To A File At Another Node
* Remote Database Access
* Program-To-Program Communication Via Sub Tasking (i.e. Father at one node and son at another)
* Program-To-Program Communication Via Message (MSG) Files (i.e. Store and Forward, similar to UNIX "pipes")
* Support PAD connected terminals in character mode.
* Support 2622's, 2624's & 2627's with VPLUS via PADs

Let us review the capabilities of DSN/DS. The slide above gives the major functions supported. The last two functions

X.25 support allows a single DS monitor program to simultaneously support all cpu-to-cpu and terminal-to-cpu activity. Up to 127 concurrent users of the "network" can enter/leave a CPU via the one INP and associated DS monitor program. So, internal CPU overhead goes down and only one modem services both types of users.

X.25 has vastly superior error recovery capability compared to the bi-sync product. Terminals may cross the continent for costs in the $6-$10 per hour range when connecting to the HP3000 via a public packet network. Of the three major U.S. domestic X.25 networks, Uninet seems to have the lowest prices, but Telenet appears to have the fastest thru-put.

HEWLETT PACKARD

Question: How can I ensure secutiry over who accesses my CPU via the X25 network?

Answer: There are two concerns:

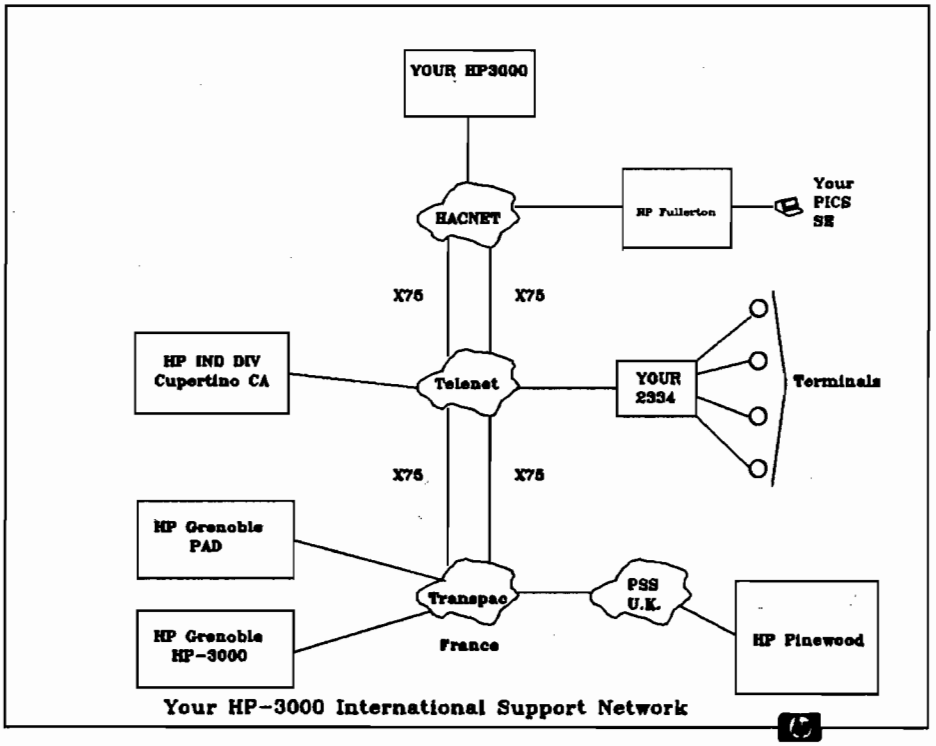
1) **CPU-to-CPU is handled by a security/routing table called NETCON. This table can only be changed by MANAGER.SYS. Two CPU's may only DS to each other if each CPU's NETCON has the other CPU in it.**
2) **Terminal-to-CPU is handled in one of several ways:**
   a) **Private Virtual Circuits (PVC's) connect one terminal to a single port of a single computer. This is just like a leased line.**
   b) **Closed User Groups (CUG's) the X25 network can have a group of PAD's and HOST's be limited to only talking to each other.**
   c) **Password on the USERID and/or ACCOUNT ID**
   d) **The HP2334 local user group feature allows network id's contained in an internal table to validate incoming calls to the PAD.**

X.25 DSN/DS allows additional security to be implemented over what a user with dial modems has with the bi-sync based product. When an X.25 network allows an HP3000 to be connected, that connection is via a leased (dedicated) circuit. The network access port to which the leased line is attached has a network controlled X.121 address.

When a CPU places a call through the network, the network ensures that in the call request packet the correct calling DTE's network ID is properly coded.

The receiving host does a table lookup in the IMAGE database named NETCON (network configuration) and will only accept a call request packet if the calling DTE's ID is in NETCON. So, CPU-to-CPU access is much tighter with X.25 than it was with bi-sync.

HEWLETT PACKARD



Your HP-3000 International Support Network

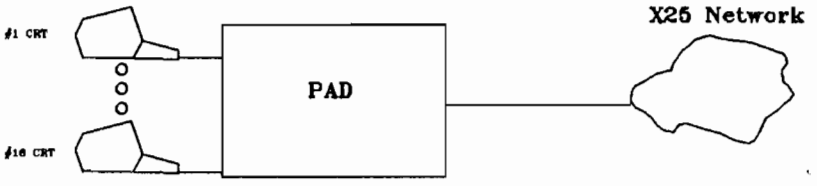
Here is an example of how four packet networks interconnect five HP3000's and two HP2334 PAD's into a "sub-network". The HP customers on a public network can be accessed by their HP branch office PICS SE for direct ONLINE support. The PICS SE can access software division cpu's in this and other countries, using DSCOPY to bring new software fixes from the divisions to your CPU in real time.

We feel it is feasible for you to discover a compiler bug, have it verified by a PICS SE, and have a later version compiler installed on your CPU, all in a matter of a few hours.

The support potential of such networks is truly exciting.

HEWLETT PACKARD

PAD = Packet Assembler Disassembler



A PAD Is: A Protocol Convertor
A Multiplexor

A PAD's benefits Are: Hi-Speed Terminal Operation
Error Free Data Transmission
Host Switching

PAD's can have modems on their terminal ports or leased line tail circuits.

The HP2334 Packet Assembler Disassembler (PAD) is a multiplexor, speed convertor, protocol convertor that allows up to 16 devices at a remote site to enter the network over one modem link. The usual crossover point economically to justify a PAD is three or more devices at a site each using the network for about 4 hours per day each.

The HP2334 is manufactured in HP's Grenoble France division. As you saw from the previous slide, the U.S. support sites have all day every day access to the Grenoble support computer to pick up new firmware revisions that can be burned into EPROM's at your U.S. support site.

Grenoble support engineers are usually able to duplicate customer problems without traveling, once they have been given a play script of a potential problem and temporary access to your X.25 host. Once they can duplicate a problem, the fix is not long in being DSCOPY'd to your U.S. support site.

HEWLETT PACKARD

The HP-2334 PAD (from Grenoble)

* Supports 4, 8, 12 or 16 Ports
* Supports up to 19.2KB from terminal to PAD
* Supports up to 19.2KB from PAD to Network
* Z80 CPU for every four terminal ports
* 16 port PAD has 5 Z80 CPU's, 72K ROM and 40K RAM
* Mnemonic Connect Table
* Closed User Group Table
* PVC Support
* Incoming SVC Support (i.e. two way SVC's)
* 25 Private User Facilities To Implement HP CRT Block Mode Support

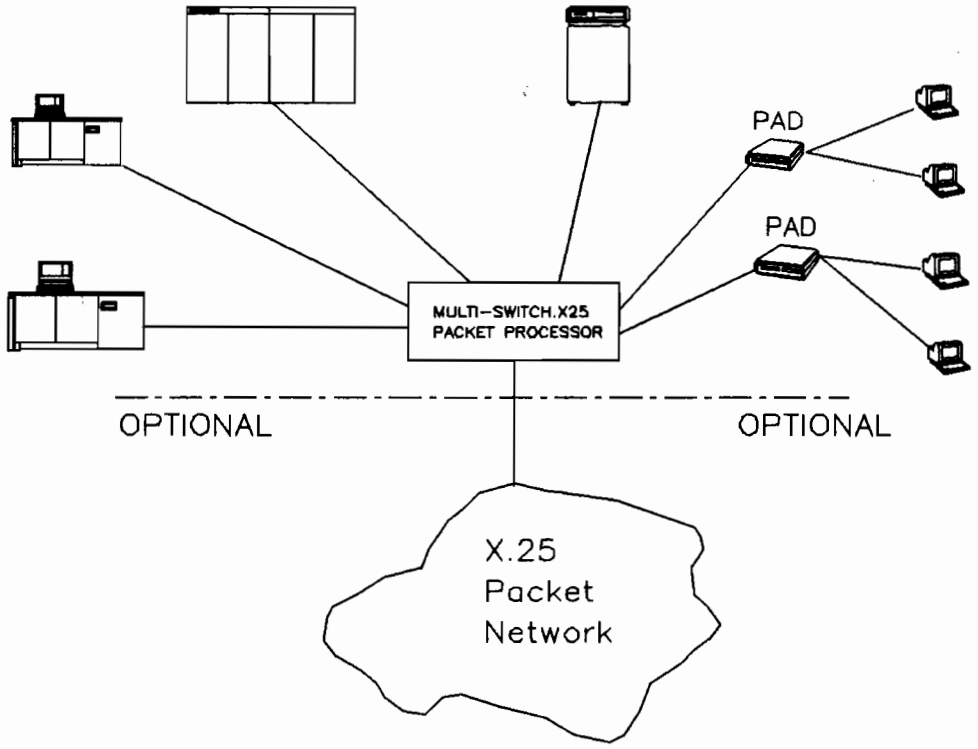
This slide outlines the main features of the HP2334 PAD product. Note that a 16 port PAD has not less than five microprocessors. The HP2334 supports a Mnemonic Connect Table. Instead of having to remember an X.121 address of up to 14 numeric digits, you simply type a number sign (#) followed by up to an 8 character name.

Any HP character printer can be connected to the 2334 and receive printout from multiple remote HP3000's. The MPE file system can now build a switched virtual circuit to the 2334 port at the time of an FOPEN, and disconnect at the time of an FCLOSE. This basic facility allows public domain programs such as RSPOOL to drive 2334 connected printers as spooled devices.

The Closed User Group table permits the 2334's support personnel to limit what remote nodes are connected to, or that may access the devices on the 2334.

The HP2334 is the hardware around which full VPLUS support of all terminals will be based. As each future MIT of MPE is released, the number of terminal types and software supported will increase.

HEWLETT PACKARD



Dynatech Packet Technology Corporation of Alexandria, VA makes a (approximately) $6,000 packet switch that from casual observation is the size and appearance of an 8 port statistical multiplexor. This packet switch can sustain roughly 34 KBS aggregrate throughput. You can run leased phone lines from each packet switch port to either a CPU or PAD and operate your own private packet switched network. Data lines should not exceed 9.6 KB on any port.

By hooking a public packet network access line to one of the packet switch's ports, you have created a `gateway'. Any PAD or CPU in your private network may now access the public network. The access facility (typically $1,500 per month) is shared.

Any DTE in your private network is accessible from the public network. Any DTE in your private network may access the public network. All of your DTE's share the cost of the one access facility.

Multiple "Multi-Switch X.25" units can connected to form larger private networks, including parallel trunks between the same two switches for greater tandem switching speed.

HEWLETT PACKARD

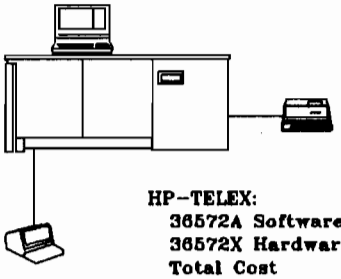
The Memotec 2500 Packet Processor is another private network/gateway product similar to the Dynatec product. The MPAC-2500 has roughly an 84 KB aggregate thruput, and will support hosts/pads up to 19.2 KB in speed. Memotec is headquartered in Toronto, with US operations based in Denver and Los Angeles (Yorba Linda suburb).

The two packet processors mentioned are just representative of many coming onto the market in 1984. Both vendors mentioned have larger switches in design, and many other vendors are introducing X.25 switches this year.

An X.25 packet processor allows you to implement a wide area network where any cpu talks to any other cpu (concurrently) with only the cost and overhead of a single datacomm controller card (INP) and a single software monitor (DSN/DS).

HEWLETT PACKARD

HP-TELEX WAS INTRODUCED IN JULY '83



Up to 10 hardware interfaces can be controlled by the one copy of software.

HP-TELEX:
36572A Software - $3,000
36572X Hardware - $2,500
Total Cost - $5,500

HPTELEX is a combination of hardware and software that allows many concurrent users of your HP3000 to both send and receive Telex and TWX traffic from around the world. HPTELEX is a product of HP Pinewood (UK)

Any text preparation package on the HP3000 can be used to prepare a Telex message. HPTELEX can be told when to dial the receiving Telex station, so as to minimize dial cos

Western Union, ITT and RCA are supported in the U.S. British Telecom is supported in the U.K., and other countries are being added.

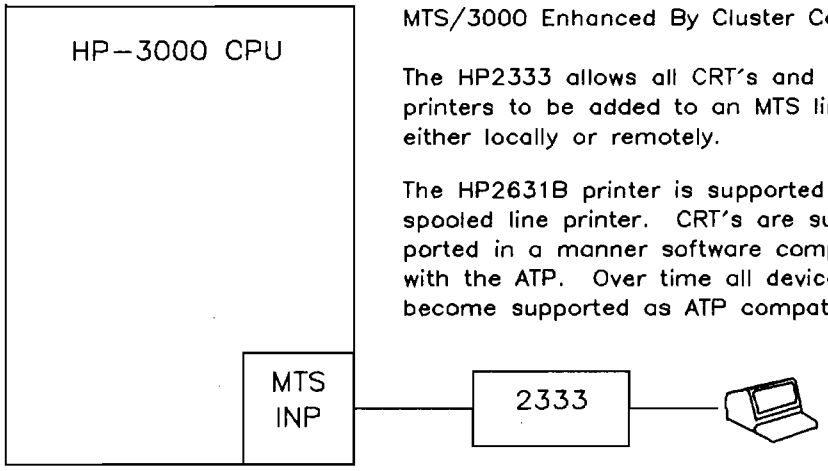
Self paced instruction leads the user through simple menu driven access to HPTELEX from any block mode terminal. Documents created by EDIT/3000, HPWORD, HPSLATE and/or TDP can be mailed with HPTELEX.

Both incoming and outgoing Telexes are queued. The queueing facility allows other users to schedule transmissions while the hardware is busy sending an earlier message. Many users can be concurrently preparing text for a single interface to transmit.

For outgoing traffic HPTELEX does all dialing, answerback code generation and verification. If a destination number is busy then HPTELEX will keep re-trying at intervals determined by the sender.

Statistics are collected and logged for all traffic. All incoming and outgoing traffic is logged, including failures and re-tries. The text of all incoming and outgoing traffic can also be conveniently reprinted or archived.

HEWLETT PACKARD



MTS/3000 Enhanced By Cluster Controller

The HP2333 allows all CRT's and character printers to be added to an MTS line either locally or remotely.

The HP2631B printer is supported asa spooled line printer. CRT's are supported in a manner software compatible with the ATP. Over time all devices will become supported as ATP compatible.

(1) The HP2333 is supported at speeds up to 19.2 KB between the INP and cluster controller, as well as between the CRT and controller. Series II/III's get relief from the 2.4KB ATC barrier.

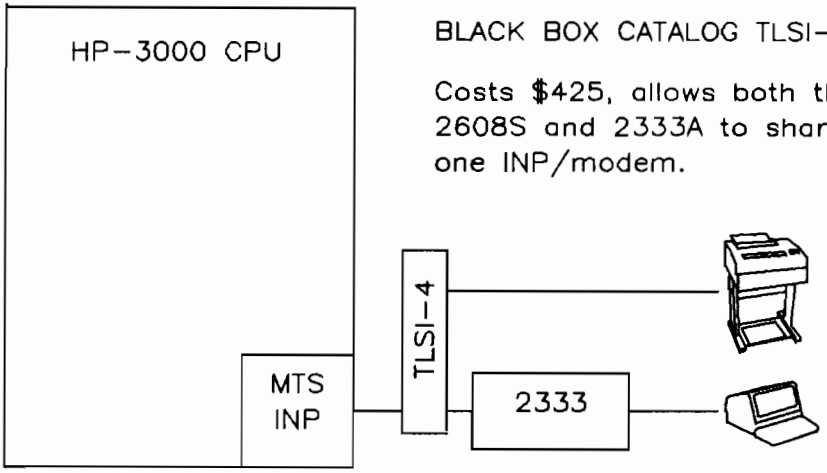
(2) The HP2333 may be either hardwired or connected via modems. Besure to use the 31390A Modem BypassCable. The 13232U cable has to have pins 15 and 24 jumpered at both ends.

The HP2333 is a new product that brings with it a new software direction for MTS/3000. HP is commited to making MTS/3000 as close in function and software compatibility with the ATP controller as is possible.

The HP2333 offers a very worthy option to Series II/III owners to greatly speed up terminal I/O and possibly reducing CPU overhead. We know that when a Series II/III is handling 2000 character interrupts per second the cpu is 100% consumed by cycle stealing! Stop and think of the impact of only four spooled 2631B's on a Series II/III. Four printers at 1200 bps each is 4800 bps aggregate or 600 chars/sec. 30% of the cpu is devoted to driving those four 2631B spooled printers!

Put the same four printers on an MTS line with a 2333 and the overhead will drop to 10-12%. Now, however, all of the CRT's on the line can operate at 9600 bps. This is very important when running VPLUS, or other terminal block mode, applications.

HEWLETT PACKARD



BLACK BOX CATALOG TLSI-4

Costs $425, allows both the 2608S and 2333A to share one INP/modem.

(1) The modem sharing device allows four remote controllers to share one modem.

(2) The INP-2333A line can run Full Duplex.

(3) On a 9600 bps line expect 200 LPM printing with 1-3 sec CRT delays.

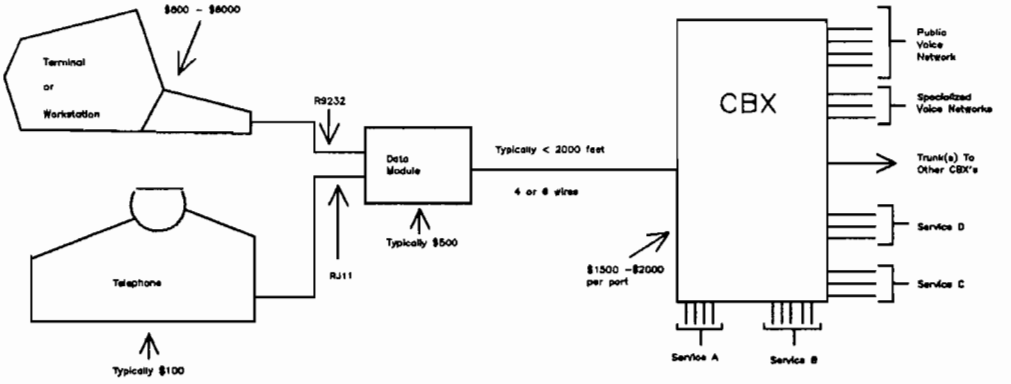
When you need both a cluster of terminals and a remote line printer at a remote site, or in the same complex, you may connect a modem sharing unit to the remote modem and then plug in up to four control units. Both the HP2333 and HP2608S want to act as control units. They both have a 25 pin male connector to plug into a modem. The TLSI-4 allows both controllers to share the one modem.

Now, the further away from the computer center the remote site is the more favorable the economics of this type of configuration become.

The HP2631B spooled printer is really only good for about 400 pages per day. An HP2608S costs about 2.5 times an HP2631B. So, about 1000 pages per day is the economic cross-over of one 2608S versus two 2631B's.

The 2608S will print at an honest 200 LPM without degrading the CRT performance more than two seconds. That is, if your CRT gives you 3 second response without an HP2608S, then you will get roughly 5 second response when the printer is running.

HEWLETT PACKARD



PBX's use a data access module to connect both a terminal and phone from a desk to the PBX.

You may have up to 4000 feet of wire from your data module to the PBX.

Four standard telephone wires are used to connect the data module to the PBX.

Norther Telecom PBX's require six wires from the data module to the PBX.

The PBX is connected to "services" such as an X25 PAD, a pool of auto-dialer modems, or a local HP-3000.

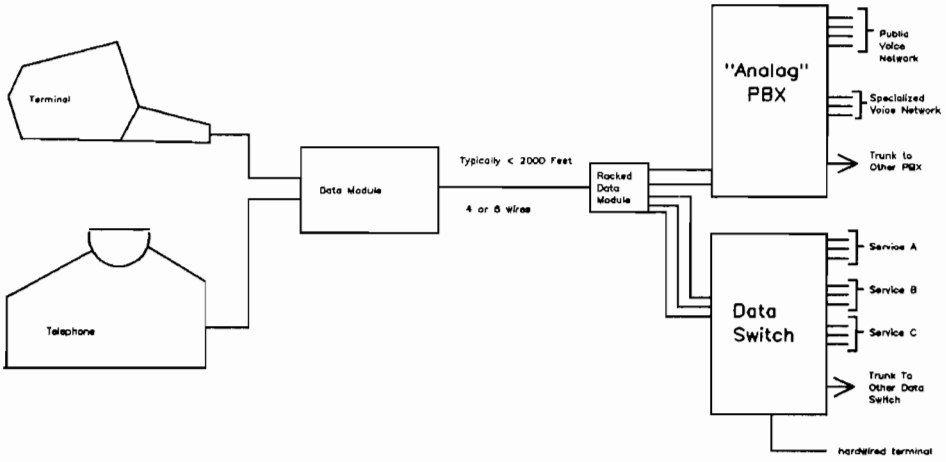
PBX's permit terminals to switch from 'service' to 'service' at 9600 bps, and to concurrently use your phone.

In early 1983 HP announced a cooperative support agreement with the PBX vendors Northern Telecom, Rolm and Intecom. The agreement calls for the vendors to work together whenever common customers are having interfacing problems between the two product lines.

A PBX appears to the HP3000 as a 9600 bps dial-up modem. Each CRT is hardwired to the PBX at 9600 bps. Some vendors also support 19.2 KB switching. The user gets the benefit of hardwired speeds and the flexibility of dialing.

Any time you have more than one computer a workstation would like to access, the PBX approach makes sense. You can place sharable autodial modems on the PBX to be shared by the hardwired CRT's. PAD's to an X.25 network are another common "service" connected to PBX's.

HEWLETT PACKARD



A 'data switch' can be used in conjunction with an older 'analog' PBX.

Data switches in combination with an analog PBX are called 'hybid' systems.

Data switches can be added to olderPBX installations to modernize them.

As with digital PBX's, your phone and terminal can be simultaneously used.

Data modules permit your phone wiring to carry both voice and data.

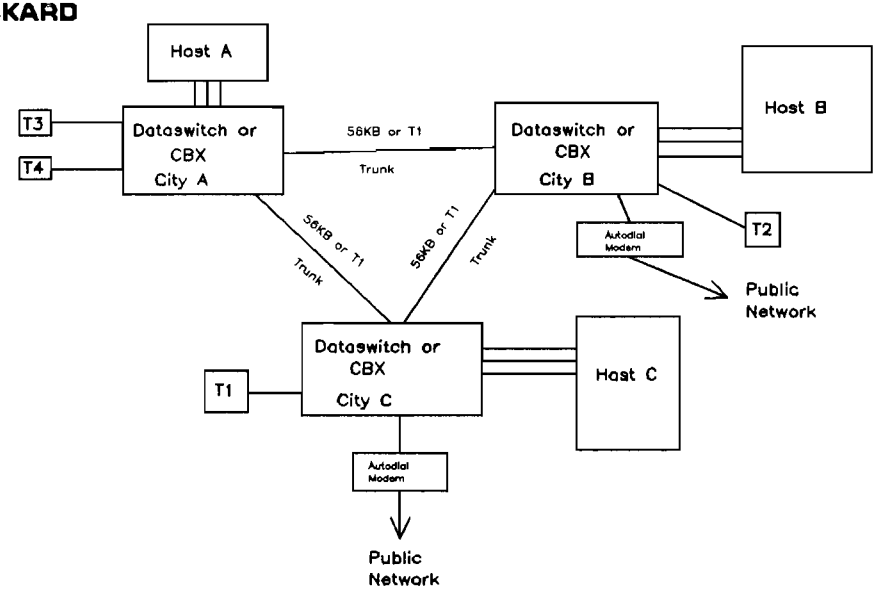
If you have an older analog PBX that will not be amortized for several years, a number of HP customers are enhancing their older analog PBX's with digital switches to produce a hybrid switched network. A number of these customers make a point of seperating their voice switch (PBX) from data switch (PABX) for reliability reasons. They are more comfortable not putting all communication eggs in one basket.

Vendors such as Infotron, Gandalf, Micom and Develcon have successfully been added to older style analog PBX's.

The "data switch" vendors allow you to install two "data modules" per workstation if the cost of running terminal data wiring is prohibitive. The data modules multiplex your desk phone and data terminal over the two pairs of telephone twisted wiring. One data module goes on your desk and one by the PBX. RS-232 wiring goes from the PBX data module to a regular computer port.

It appears to be about a 50/50 split amongst HP's customers when we ask whether customers are planning to use local area video networks or PBX's to connect workstations to multiple servers (cpu's, print stations, electronic filing cabinets).

HEWLETT PACKARD



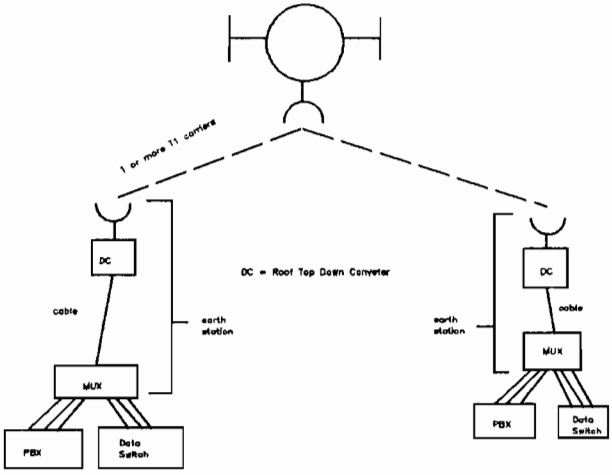
CBX's permit trunk lines to iterconnect all other CBX's into on integrated voice/data network. Capitalization can be as high as $2,500 per port. Terminals1 thru 4 may switch to ony host using their keyboard number pods and still operate at 9.6 KB.

CBX's, PBX's and data switches all seem to permit a "trunk" line to be run between each other. These "tie lines" allow both voice and data switching between sites to occur.

Terminals at one site may now connect to services at all PBX sites.

Companies with enough voice and data volume between two sites should give serious consideration to "trunking" their PBX/CBX's. Most planners feel that when the total phone bill between two sites is about 50% more than the cost to lease a trunk line, then the crossover point has been reached. The capital and administrative costs of upgrading the PBX's accounts for the need to be 50% over the cost of a simple leased line.

HEWLETT PACKARD



Major organizations may completely bypass the telco for all traffic between their own sites by using satellite circuits and their own earth station equipment. A complete earth station costs about $250,000. But, costs are comming down at a 30% per year rate. This sort of trunking will be very common by 1985. The dish antennas are about 10 feet in diameter.

Satellite circuits are approximately half the price of a similar land based service. The satellite circuit typically has about 60% of the data capacity of a land line.

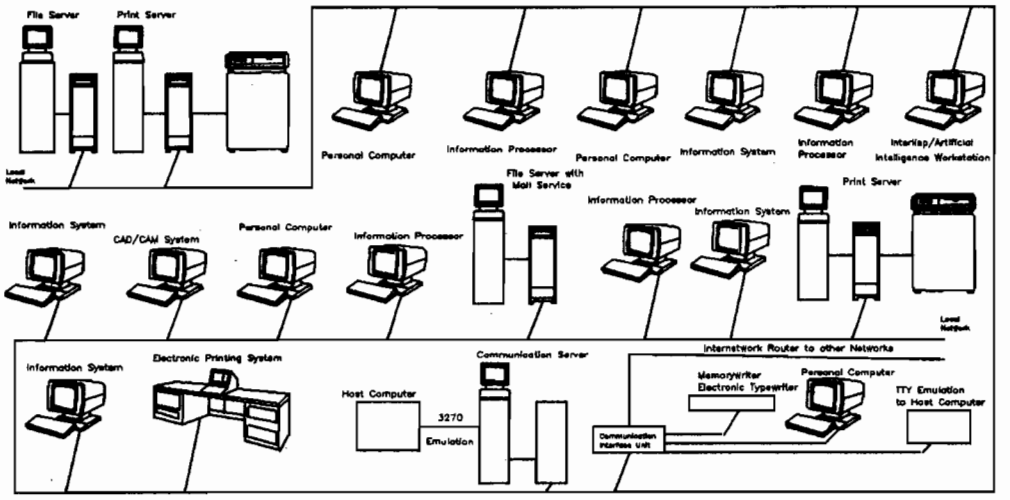
Protocols such as HDLC and SDLC are full duplex in nature and allow for a number of unacknowledged buffers to be sent before the sending DTE stops and waits for acknowledgement of error free receipt. Data can be going in both directions concurrently.

HP3000 users of remote DS/3000 may want to switch to the X.25 based DS/3000 over leased satellite circuits. While individual data streams may run somewhat slower, aggregate thru-put should increase. The big thing to remember is the lower individual data stream rates are offset by an even lower decrease in datacomm cost.

Summarizing, the X.25 protocol allows multiple concurrent DS users to achieve better link usage than the bisync protocol. Satellite circuits will be slower than land lines, but at a better price performance ratio.

HEWLETT PACKARD

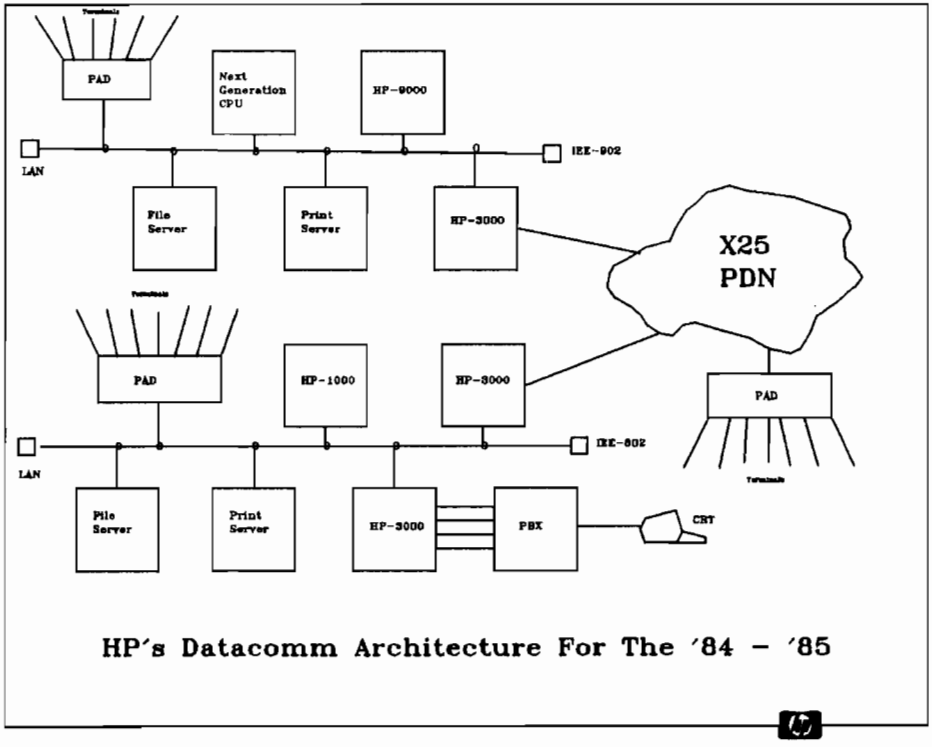
A HYPOTHETICAL LOCAL AREA NETWORK



Switching the topic to Local Area Networks (LAN's) this slide shows a hypothetical network. File servers allow common data to be shared among the individual nodes on the video cable network. High cost disks with backup facilities may be shared across multiple workstations and be very cost competitive to personal Winchester disk drives at each station. The backup facilities cannot be over emphasized.

Print server's allow an expensive printer, such as a fancy laser printer that can mix diagrams, words and graphs on one page, to be shared by enough workstations that individual daisy wheel printers become archaic. Laser print servers can print multiple spooled copies of documents while the individual workstations proceed to do other work.

HEWLETT PACKARD



HP's Datacomm Architecture For The '84 - '85

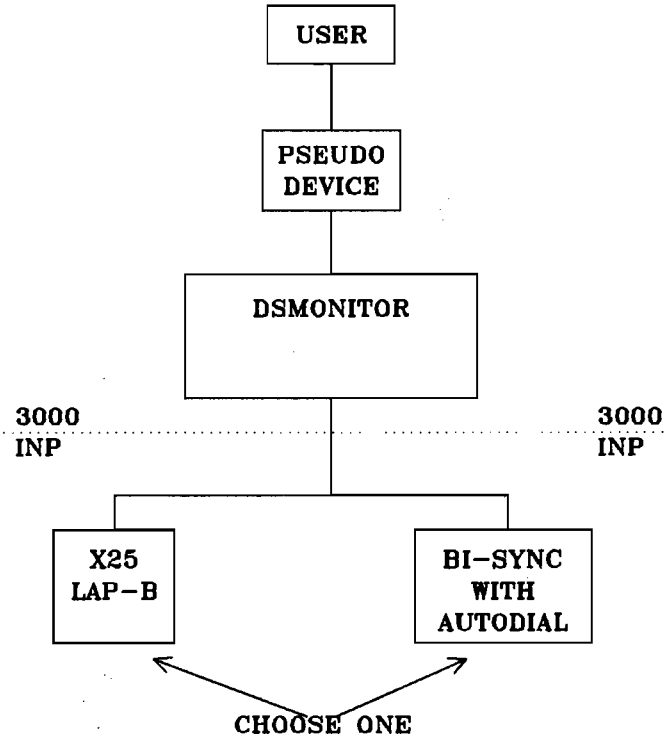
This very busy slide shows partially where HP is, and the rest shows where we are going. Today, HP supports both X.25 packet networks and PBX's. Sometime in 1984 we will be adding support of IEE-802 Local Area Networks (LAN's). LAN's will need a PAD product to economically connect multiple terminals to the cable network.

Once a LAN is introduced it will become the backbone of HP's MPN architecture. Personal workstation products will be connected to cpu's, shared database nodes and printer nodes via LAN's.

Communication within an area of several adjacent buildings will be via LAN's. Individual LAN's (subnets) will be networked together via "gateway" cpu's and X.25 packet networks.

You will see LAN PAD's and at least one file server and one print server product from HP in 1984. Initial CPU support will include HP3000, HP1000, HP9000, HP210 and the Shared Resource Manager. In addition, the HP200 and HP100 products will have a low speed LAN (i.e. 1 MB/sec) and file server, with gateway products for the IEE-802 and other communication services.

HEWLETT PACKARD



The last two slides I want to present go into the architecture of DSN/DS, as it is today and as it will be in 1984. Clearly HP is moving towards the ISO Open System Reference Model of seven layer software.

This slide shows how the present DSN/DS (which we shall call DS 1) is structured. A pseudo device is used for interprocess communication between a program and the DS monitor program (DSMON).

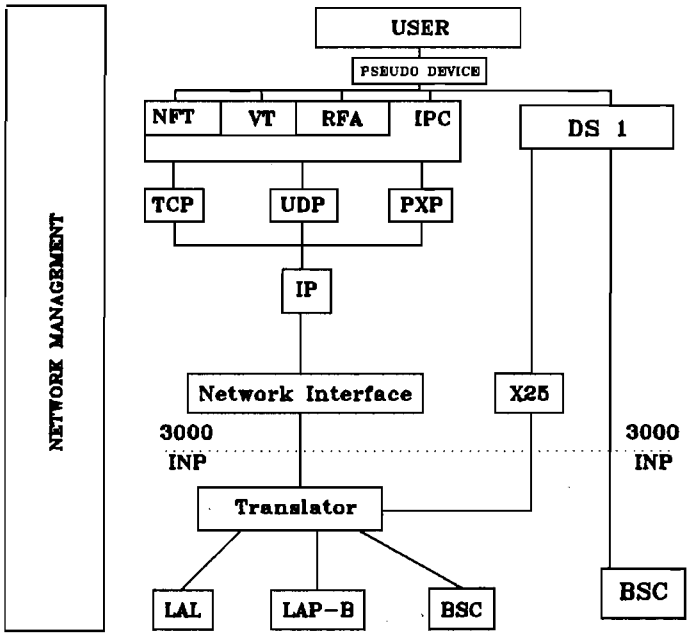
The monitor program downloads an INP with software layers 1 and 2. The monitor is layer 3. Layers 4 thru 7 run on the user stack as extensions of the MPE file system.

The architecture of DS-1 does not permit such niceties as: store and forward networking; alternate path routing; transparent pass- thru of intermediate nodes.

And so ..... we have come up with an architecture for DS-2 (that is an internal name) which greatly enhances DSN/DS.

HEWLETT PACKARD

ARCHITECTURE OF NEXT GENERATION DSN/DS



This slide shows how DS-1 will be integrated into DS-2. The present HP-32190 product (or its replacement HP-32189/HP-32191 products) will be able to co-exist with the new architecture. But DS-2 introduces several major new software components:

(1) The Network Interface or "transport" layer will allow a circuit to come in from a LAN and go out on a X.25 or Bi-sync link. This pass-thru will not require a session to be started on the local HP CPU.

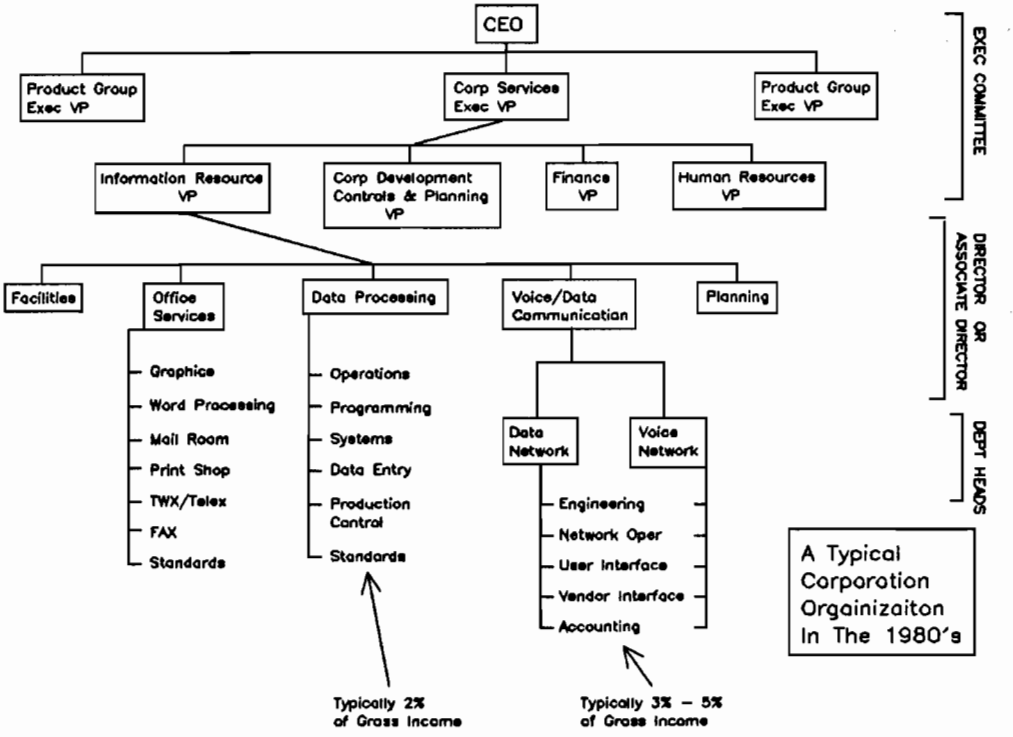
(2) The Arpanet level 4/5 protocols known as TCP/IP will be the HP level 4/5 protocols. Network Interface expects to see TCP/IP protocol for activity originating above it, or destined for layers above it.

(3) Alternate path routing when one link fails.

(4) Extended error recovery so that when a remote session becomes unreachable temporarily that traffic for it is stored, and then forwarded when the link can be reestablished.

You will see this architecture in the DSN/DS that supports local area networks.

HEWLETT PACKARD



When we look at the sum of our installed base, HP sees its customer organizations reorganizing to reflect the span of control needed as companies really move into distributed computing. The data and voice networks of organizations are being merged to allow common management.

This slide presents what we feel is a trend in customer organization structure. In "Information Resource" executive is routinely being given control of the "network" and administrative organizations that most heavily use and/or administer the network.

More organizations now merge the voice and data network management than do not.

HEWLETT PACKARD

NOTES:

STOP GROWING MUSHROOMS PART III

Tom Knight
Hewlett Packard

In my paper presented at the scrug conference last year I stated that I felt it was all to easy to create problems in your operations department by not keeping your operators informed. Stop growing mushrooms, "keep them in the dark and feed them a lot of *fertilizer".

Thanks to my friend Marc Covitt, we have updated that phrase to "keep them in the dark and feed a lot of *fertilizer and if they grow or prosper, then cut them off at the knees".

These two phrases are some what the same as saying "what they don't know will not hurt them". So again I will try to provide some helpful information in the operations area.

We will spend some time with:

Operations Guidelines

Run Documentation

UDC's and other tricks

Hints on how to keep your operators happy

* not quite the word normally used.

Operations Guidelines

This is an area that most shops really seem to be lacking in. What you'll find is a total lack of guidelines or very poor guidelines.

You can't expect your users to work well with you if you don't provide them with information on what you expect of them.

Example 1:

```
:FILE T;DEV=TAPE
:STORE PINC0@.DATA.PROD;*T;SHOW


:TELLOP ;
:TELLOP ; ************************************
:TELLOP ; * MOUNT 2400' SCRATCH TAPE.        *
:TELLOP ; * LABEL IN TAPELIB AS PIN0001T.    *
:TELLOP ; ************************************
:TELLOP ;
:FILE PIN0001T;DEV=TAPEOUT;DEN=1600
:RUN TS0065P.UTIL.SYS;INFO="PINC0@.DATA.PROD;*PIN0001T; &
:      SHOW;CONSOLE;CHECK=10"
```

Note

TS0065P is a utility program written by Joe Fleming at Hewlett Packard San Diego Division which is safe to run (runs in user mode). After checking for the extended keywords, the STORE COMMAND is passed to MPE via the COMMAND INTRINSIC. This Program will be on the latest SWAP Tape, with two versions which will allow it to run on either Ciper or Q-Mit.

Both methods work, and will be accepted by MPE but, you can bet the second method will work far better and create less chances for something to go wrong.

Example 2:

```
:FILE PRT;DEV=LP;CCTL

:COMMENT ; ********************************************
:COMMENT ; * REPORT PIN2202R CREATES OVER 12000 SECTORS *
:COMMENT ; * OF SPOOL FILES. PRINT AT YOUR CONVIENCE ON *
:COMMENT ; * ON STANDARD 1-PART PAPER   8 LPI.          *
:COMMENT ; ********************************************
:COMMENT ;
:FILE PIN2202R;DEV=LP,7,1;CCTL
```

In both examples, it's plain to see that you can force your operators into their mind reading act or you can inform them of what's supposed to happen.

Tell

Tell them what you expect.
Tell them how to do it.
Tell them if they did it right.

Sample JCL Implementation Plan

I. Prepare JCL as per New Standards
   A. USE file (see Use file documentation)
   B. TDP or other means

II. Submit JCL "Package" to operations for review
   A. hardcopy JCL listing
   B. move file (if needed)
   C. run documentation (as needed - per operations request)
   D. large jobs (require 5 day minimum in advance)
   E. small jobs (require 2 day minumum in advance)

III. Acceptance/Rejection
   A. JCL listing returned to analyst
      1. acceptance
      2. rejection and reasons why
      3. analyst makes corrections and resubmits
         a. gets higher priority

IV. Exceptions
   A. "HOT" production jobs
      1. accepted immediately after change
      2. PR generated to bring JCL to standards

V. Key JCL for Renovation
   A. based on top percentage of problem reports (PR's)
   B. 1 JCL/group/week
   C. all JCL changed as problems occur
   D. all JCL related to any MR

All stages are required except those specifically noted as optional.

Stages 1-7 and 21-27 are Job related.
Stages 8-20 are step related.

Stage	Command	Explanation
1	!JOB	Conform to naming standards must be the same as the JCL file
2	!COMMENT	1-3 lines describing the job.
3	!COMMENT	Frequency daily/weekly/monthly
4	!COMMENT	Run time 10 minutes, 2 hours, etc.
5	!COMMENT	Author and date written
6	!COMMENT	JCL revision history: name, date, brief description
7	!COMMENT	All input files and data set/base names and their source
8	!TELLOP !TELLOP;********** !TELLOP;* JOB= jobname step=step-no phase=program * !TELLOP;* description of the step and/or operator action * !TELLOP;********** !TELLOP	
9	!SHOWTIME	
10	!SHOWJOB	
11	!RESET@	
12	!SETJCW	Optional
13	!PURGE	Purge a file before attempting a !BUILD
14	!COMMENT	A brief description of why the following !BUILD statements are not blocked the same as the BLOCK program recommends
15	!BUILD	For all output disc files !BUILD FILE XX;DISC=NN,32,16;ETC where NN = 2 times the antipated maximum volume of records. If this is a temporary step related work file, then use ";TEMP"
16	!FILE	Required for all files of any file type for each step. For output disc files use: !FILE FILE-a=FILE-x,OLD or OLDTEMP to reference the previous !BUILD
17	!COMMENT	For restart instructions

```
18              !CONTINUE         Optional
19              !RUN Program
20              !IF               Optional

21              !LISTF listf,2    For all output disc files

22              !PURGE            For input and output files not
                                  required for another job

23              !COMMENT          Used if a file is not purged,
                                  give reason why it was purged

24              !RUN SCRUNCH      For all output disc files not purged

25              !COMMENT          Used if a file was not SCRUNCHed

26              !TELLOP;
                !TELLOP; ******************************************
                !TELLOP; *     ///// jobname  IS  E  O  J /////     *
                !TELLOP; ******************************************
                !TELLOP;

27              !EOJ
```

Example 3:

A sample JCL using this standard

```
!JOB ICA2002J,JOBS.PROD,ICAPPS
!COMMENT ********************************************************
!COMMENT         JOB DESCRIPTION
!COMMENT ICAPPS DAILY BATCH REPORTS, INCLUDING WORKORDER DOC.
!COMMENT
!COMMENT RUN FREQUENCY:  DAILY
!COMMENT JOB TAKES APPROX.:  1:30 HRS.
!COMMENT WRITTEN BY: DORENE MATNEY    02-18-83
!COMMENT ********************************************************
!COMMENT        JCL REVISION HISTORY      (NAME, DATE, DESCRIPTIO
!COMMENT
!COMMENT   02/23/83   DORENE MATNEY   (M968)
!COMMENT      -ADDED STEPS 062, 064, 066, 068, 069 TO PRINT
!COMMENT       MATERIAL LISTS AS PART OF WORKORDER PACKET
!COMMENT      -ADD FILE MRF0156D FROM WOF0230P AS MATERIAL
!COMMENT       LIST REQUEST FILE
!COMMENT ********************************************************
!COMMENT     INPUT FILES    USED BY THIS JOB AND THEIR SOURCE
!COMMENT MRF0152C (CONTROL CARD USED BY MRF1100P--FROM SADIE)
!COMMENT MRF0700P, MRF1100P, MRF0940P (FROM MRF0001J--SADIE)
!COMMENT WOF2300D (ICA2001J)
!COMMENT WOF0014D
!COMMENT XEQST503
!COMMENT ICA3020D (ICA2001J)
!COMMENT PINCO
!COMMENT MRFBAS
!COMMENT ICAPPS
!COMMENT
!TELLOP *=======================================================
!TELLOP *   JOB=ICA2002J  STEP=010  PROGRAM=ICA2010P.PROG
!TELLOP * STRIP CURRENT CONTROLLER ASSIGNMENTS
!TELLOP *=======================================================
!SHOWTIME
!SHOWJOB
!RESET @
!COMMENT
!COMMENT
!COMMENT
```

```
!   FILE PINCO.PROD.DATABASE=PINCO.DATA.PROD
!COMMENT ************************************************************
!COMMENT *   RESTART INSTRUCTIONS:
!COMMENT * RESTART AT THIS STEP
!COMMENT ************************************************************
!   RUN ICA2010P.PROG;LIB=G
!COMMENT
!TELLOP *=============================================================
!TELLOP *   JOB=ICA2002J  STEP=020  PROGRAM=FCOPY.PUB.SYS
!TELLOP * LOADS PINCO GENERATED RUNS FROM TAPE TO DISK
!TELLOP * ------------------------------------------
!TELLOP * MOUNT TAPE "PIN4603T" CREATED IN PINCO TONIGHT
!TELLOP *              (1600 BPI)
!TELLOP *=============================================================
!SHOWTIME
!SHOWJOB
!RESET @
!   PURGE PIN4603D
!COMMENT
!   BUILD PIN4603D;REC=-80,144,F,ASCII;DISC=1584,12,6
!   FILE PIN4603D,OLD
!   FILE PIN4603T;DEV=TAPEIN;REC=-80,10,F,ASCII
!COMMENT ************************************************************
!COMMENT *   RESTART INSTRUCTIONS:
!COMMENT * RESTART AT THIS STEP
!COMMENT ************************************************************
!   RUN FCOPY.PUB.SYS
FROM=*PIN4603T;TO=*PIN4603D;SUBSET
EXIT
!COMMENT
!TELLOP *=============================================================
!TELLOP *   JOB=ICA2002J  STEP=030  PROGRAM=SORT.PUB.SYS
!TELLOP * SORT TRIGGERS FOR WORKORDER PACKET
!TELLOP *=============================================================
!SHOWTIME
!SHOWJOB
!RESET @
!   PURGE TRIGSORT
!COMMENT
!COMMENT
!   FILE INPUT=WOF2300D,OLD
!   FILE OUTPUT=TRIGSORT
!COMMENT ************************************************************
!COMMENT *   RESTART INSTRUCTIONS:
!COMMENT * RESTART AT THIS STEP
!COMMENT ************************************************************
!   RUN SORT.PUB.SYS
KEY 1,12
END
!   PURGE WOF2300D
!   RENAME TRIGSORT,WOF2300D
!COMMENT
!TELLOP *=============================================================
!TELLOP *   JOB=ICA2002J  STEP=040  PROGRAM=WOF0230P.PROG
!TELLOP * FORMATS WORKORDERS FOR PRINTING
!TELLOP *=============================================================
!SHOWTIME
!SHOWJOB
!RESET @
!   PURGE WOF2301D
!   PURGE MRF0156D
!COMMENT
!   BUILD WOF2301D;REC=-94,128,F,ASCII;DISC=48000,32,16
!   BUILD MRF0156D;REC=-80,160,F,ASCII;DISC=3040,20,10
!   FILE WOF2301D,OLD
!   FILE MRF0156D,OLD
!   FILE WOF2300D,OLD
```

```
!   FILE PIN4603D,OLD
!   FILE PINCO.DATA=PINCO.DATA.PROD
!   FILE MRFBAS.DATA=MRFBAS.DATA.PROD
!COMMENT ********************************************************
!COMMENT *   RESTART INSTRUCTIONS:
!COMMENT * RESTART AT THIS STEP
!COMMENT ********************************************************
!   RUN WOF0230P.PROG
!COMMENT
!TELLOP *=======================================================
!TELLOP *   JOB=ICA2002J  STEP=050  PROGRAM=WOF0240P.PROG
!TELLOP * PRODUCES HARDCOPY WORKORDERS
!TELLOP *=======================================================
!SHOWTIME
!SHOWJOB
!RESET @
!   PURGE WOF0017D
!COMMENT
!   BUILD WOF0017D;REC=-82,128,F,ASCII;DISC=5120,21,11
!   FILE WOF0017D,OLD
!   FILE WOF2301D,OLD
!   FILE WOF0014D,OLD
!   FILE WORKORDS;DEV=FORMS,6,1;CCTL;FORMS=MOUNT WORKORDER FORMS
!COMMENT ********************************************************
!COMMENT *   RESTART INSTRUCTIONS:
!COMMENT * RESTART  AT THIS STEP
!COMMENT ********************************************************
!   RUN WOF0240P.PROG



!   PURGE ICA0250D
!   PURGE ICA0251D
!   PURGE ICA2401D
!   PURGE ICA2401W
!   PURGE ICA3020D
!   PURGE ICA3090D
!   PURGE ICA3091W
!   PURGE ICA3092W
!   PURGE ICA3093W
!   PURGE ICA3094W
!   PURGE MRF0156D
!   PURGE MRF0700D
!   PURGE MRF0700W
!   PURGE MRF0701W
!   PURGE MRF0701D
!   PURGE PIN4603D
!   PURGE WOF0017D
!   PURGE WOF2301D
!   PURGE WOF0250D
!   PURGE WOF2300D
!COMMENT
!COMMENT
!COMMENT
!TELLOP *=========================================*
!TELLOP *     /////    END OF ICA2002J     /////
!TELLOP *=========================================*
!SHOWTIME
!EOJ
```

Run Documentation

Now here is an area that programmers have become very famous for. If given the chance most programmers would provide little or no documentation with new programs. I feel I should make it clear that most programmers because of time constraints and massive work loads would like to avoid spending time creating run documentation, but the job's not done until the paperwork is complete. Although time consuming, this can make life better for the programmers. With good JCL and QUALITY DOCUMENTATION can avoid some of those late night or weekend (vacation) calls from your bewildered operators.

COMPUTER DOCUMENTATION

PAGE 1 of 9

JOB NAME: ICA2002J PROGRAMMER: DORENE MATNEY

RUN TIME: 1 1/2 HOURS LOGON: OPS.PROD

FREQUENCY: DAILY COMMENTS: LET ONLINE USERS ON

SYSTEM: GEORGE

STEP: 010
PHASE NAME: ICA2010P PINCO
CONTROL
DESCRIPTION: STRIP MASTER

CONTROLLER ASSIGNMENTS

OUTPUT DISTRIBUTION: ICA2010P

TAPES: N/A

REPORTS: N/A WOF0014D

FICHE: N/A

STEP: 020
PHASE NAME: FCOPY PIN4603T

DESCRIPTION: LOAD PINCO

RUN DATA TO DISC

OUTPUT DISTRIBUTION: FCOPY

TAPES: N/A

REPORTS: N/A PIN4603D

FICHE: N/A

UDC's

OPSUDC1.OPS.PROD

STARTUP	USER
CMD	USER
INIT	USER
AJ	USER
JAM	USER

```
    W                   USER
    RE                  USER
    SO                  USER
    SOR                 USER
    AF                  USER
    AF19                USER
    AF6                 USER
    ASF                 USER
    R                   USER
    WEL                 USER
    HOFF                USER
    HON                 USER
    LIM                 USER
    DEL                 USER
    B                   USER
    0                   USER
    SUS                 USER
    UNL                 USER
    SYSMON              USER
    TERM                USER
    SL                  USER
    SLOG                USER

SYSTEMU.PUB.SYS
    A                   SYSTEM
    ASK                 SYSTEM
    B                   SYSTEM
    BANNER              SYSTEM
    BANNERCOB           SYSTEM
    BANNERFTN           SYSTEM
    BANNERSPL           SYSTEM
    BLD                 SYSTEM
    BLOCK               SYSTEM
    CMD                 SYSTEM
    CREATOR             SYSTEM
    DATES               SYSTEM
    DBSHOW              SYSTEM
    DBUTIL              SYSTEM
    DECOMP              SYSTEM
    DBWHO               SYSTEM
    DIR2                SYSTEM
    EDIT2               SYSTEM
    END                 SYSTEM
    FILES               SYSTEM
    FINDFILE            SYSTEM
    FORMSPEC            SYSTEM
    GRAPH               SYSTEM
    HARDCOPY            SYSTEM
    HPDRAW              SYSTEM
    HPSLATE             SYSTEM
    HPWORD              SYSTEM
    LISTFILE            SYSTEM
    OPT                 SYSTEM
    OPTICALC            SYSTEM
    PURG                SYSTEM
    Q                   SYSTEM
    QUERY               SYSTEM
    RP                  SYSTEM
    RU                  SYSTEM
    SEG                 SYSTEM
    SJ                  SYSTEM
    SJS                 SYSTEM
    SP                  SYSTEM
    SPOOK               SYSTEM
    ST                  SYSTEM
    STREAMER            SYSTEM
    TDP                 SYSTEM
```

```
   TOOLSET              SYSTEM


STARTUP
OPTION LOGON,NOBREAK
RUN ALLOWME.UTIL.SYS;PARM=1
CONTINUE
FILE STORE;DEV=TAPE
FILE RESTORE;DEV=TAPE
FILE SYSLIST;DEV=LP,7,1
RUN LISTEQ2.PUB.SYS
****
INIT
OPTION LIST
LIM
JOBFENCE 5
OUTFENCE 7
OUTFENCE 8;LDEV=19
STREAMS 10
COMMENT SYSTEM IS UP AND RUNNING
****
AJ PARM
ABORTJOB  !PARM
****
JAM PARM
SUSPENDSPOOL !PARM
OPTION LIST
CONTINUE
RESUMESPOOL !PARM
****
W PARM=0
SETJCW WARNING = !PARM
IF WARNING = 3 THEN
    WARN @;MRF STARTING -
           PLEASE EXIT THE MRF DATABASE
ELSE
IF WARNING = 5 THEN
    WARN @;NIGHTLY PRODUCTION STARTING
           PLEASE LOGOFF
    ENDIF
ENDIF
****
RE
RECALL
****
SO
SHOWOUT ACTIVE
CONTINUE
SHOWOUT READY
****
SOR PARM
SHOWOUT JOB= J!PARM;READY
****
AF PARM
OPTION LIST
ALTSPOOLFILE  O!PARM;PRI=9
****
AF19 PARM
OPTION LIST
ALTSPOOLFILE  O!PARM;DEV=19
****
AF6 PARM
OPTION LIST
ALTSPOOLFILE  O!PARM;DEV=6
****
ASF PARM PARM1 PARM2 PARM3=1
```

```
OPTION LIST
ALTSPOOLFILE  0!PARM;DEV=!PARM1;
              PRI=!PARM2;COPIES=!PARM3
****
R PARM PARM1
REPLY !PARM,!PARM1
****
WEL
WELCOME
****
HOFF PARM
OPTION LIST
HEADOFF !PARM
****
HON !PARM
OPTION LIST
HEADON !PARM
****
LIM PARM=2 PARM1=64
OPTION LIST
LIMIT !PARM,!PARM1
****
DEL PARM
OPTION LIST
DELETESPOOLFILE  !PARM
****
B PARM
BREAKJOB  !PARM
****
O PARM
OPTION LIST
OUTFENCE !PARM
****
SUS PARM
SUSPENDSPOOL !PARM
****
TERM PARM
FILE TERM,NEW;REC=-132,,F,ASCII;DEV=!PA
RUN POW0001P.JCL.TEST
****
SL
SHOWLOG
****
SLOG
SHOWLOGSTATUS
****


STARTUP
OPTION LIST,LOGON,NOBREAK
CONTINUE
LIMIT 0,0
CONTINUE
OUTFENCE 12
CONTINUE
JOBFENCE 14
CONTINUE
STREAMS 10
CONTINUE
TUNE 1000;CQ=152,200,0,300;DQ=202,238,1
CONTINUE
FILE STMFILE=SYS0200J.JCL.PROD
CONTINUE
COMMENT RUN TS0020P.UTIL.SYS;PARM=1
CONTINUE
FILE STRMFILE=SYS0001J.SUPPORT
CONTINUE
```

```
RUN STREAMX.PUB.VESOFT;PARM=1
CONTINUE
DSCONTROL HAL;OPEN,NOW
CONTINUE
DSCONTROL KERMIT;OPEN,NOW
CONTINUE
DSCONTROL GEORGE;OPEN,NOW
CONTINUE
DSCONTROL SADIE;OPEN,NOW
CONTINUE
BYE
***

!JOB SYS0001J,JOBS.PROD,EDP
!COMMENT ********************************
!COMMENT *                  JOB DESCRIPTIO
!COMMENT * This job stream is part of t
!COMMENT * procedure. A log-on UDC for
!COMMENT * essential commands, then str
!COMMENT * allocates space for system u
!COMMENT *
!COMMENT * RUN FREQUENCY:  With every s
!COMMENT * JOB TAKES APPROX.:  2 minute
!COMMENT * WRITTEN BY: Jeannine Smith
!COMMENT ********************************
!COMMENT
!COMMENT ********************************
!COMMENT *   JCL REVISION HISTORY
!COMMENT *
!COMMENT *
!COMMENT *
!COMMENT ********************************
!COMMENT

!TELLOP ********************************
!TELLOP *   JOB=SYS0001J
!TELLOP *   ALLOCATEs system utilities
!TELLOP ********************************
!SHOWTIME
!SHOWJOB
!RESET @
!ALLOCATE EDITOR.PUB.SYS
!CONTINUE
!ALLOCATE QUERY.PUB.SYS
!CONTINUE
!ALLOCATE LISTDIR2.PUB.SYS
!CONTINUE
!ALLOCATE TS0020P.UTIL.SYS
!CONTINUE
!ALLOCATE LOGON.UTIL.SYS
!CONTINUE
!ALLOCATE INTRFACE.UTIL.SYS
!CONTINUE
!ALLOCATE LISTER.UTIL.SYS
!CONTINUE
!ALLOCATE TDP.PUB.SYS
!CONTINUE
!ALLOCATE DSCOPY.PUB.SYS
!CONTINUE
!ALLOCATE SCRIBE.PUB.SYS
!CONTINUE
!ALLOCATE FCOPY.PUB.SYS
!CONTINUE
!ALLOCATE TSJ0600P.UTIL.SYS
!CONTINUE
!ALLOCATE OPT.PUB.SYS
!CONTINUE
```

```
!ALLOCATE TS0750P.UTIL.SYS
!CONTINUE
!ALLOCATE USER.PUB.SECURITY
!CONTINUE
!ALLOCATE STORE.PUB.SYS
!CONTINUE
!COMMENT
!TELLOP ********************************
!TELLOP *       END OF SYS0001J
!TELLOP ********************************
!SHOWTIME
!EOJ
```

HINTS ON KEEPING YOUR OPERATORS HAPPY!

At Hewlett Packard we use a system called Management by Objective If management has established the plans, objectives and goals of the company and department, then the individual can set his or her goals with the best interests of all concerned. Point, if management can efficiently handle:

Planning

Organizing

Controlling

Leading

Then the employees can set their goals to emphasize what should be done rather than how it is to be done, leaving room for creative solutions.

Since most EDP departments tend to run on a 24 hour 7 - day schedule, some very real problems can and most often occur. It's not unusual for an operator(s) to be be working with little or no supervision during the night shift and weekends. This makes it very important that supervision provide a constant flow of information to and from the operators. More often than not I have found that once an employee ends up on night shift, they tend to hear from supervision only when they do something wrong. When an employee hears only negative thoughts, then they tend to go astray.

One of the things that we have found at San Diego Division to help keep operators happy is to provide special projects for them to perform.

A project can range from something very simply like keeping an inventory of printer ribbons and reordering when needed, to writing complex programs. This requires that supervision spend just a little time with the operators on an ongoing basis to learn there wants, desires and needs. This time spent can provide a bonus for both the employee and the company, by providing a vehicle of upward learning for the employee and better performance which helps the company.

ZEN
AND THE ART OF
SOFTWARE MAINTENANCE

MARC COVITT
SYSTEMS ENGINEER
HEWLETT-PACKARD

PURPOSE:

This paper and the associated presentation will discuss the issues of Software Maintenance. The author will present an opinion that many types of software problems can be prevented by exercising regular preventative steps. There is an analogy to the steps taken as part of hardware maintenance. The author will also discuss several utilities available to perform the maintenance function. Specific areas covered will include:

Programming Techniques
Disc Space Management
Data Base Checking

OVERVIEW:

All of us are aware of the concept of hardware maintenance. On your computer system, your Customer Engineer will perform two types of maintenance. One is Remedial Maintenance--- in other words, something is broken and it needs to be fixed. The other is called Preventative --- nothing is broken (hopefully) and we take the time to perform routine custodial care. The purpose of preventative maintenance is to minimize the occurence of remedial maintenance and to prolong the life of the equipment.

In addition to your computer system, consider the case of your car. (Yes, I know Eugene that it will be several days before you have one--- but consider it anyway.) When something goes wrong, you take it in to be remedied. Periodically, you should also take it in for routine preventative care--- at least to change the oil and give it a tune-up.

Having introduced the simple terminology, I now move on to my first point: --

PREVENTATIVE MAINTENANCE IS NOT JUST FOR HARDWARE.

It is logical and reasonable to extend the PM philosophy to our software as well. Periodically we should make routine checks on our programs and systems as well as the hardware. After all, if an application fails to perform and is inoperative for the user, it may not matter to the user if the cause was hardware or software--- the result is the same.

With regular PM's on our hardware, we do things like change the filters, clean the equipment, and verify that the equipment performs within certain known tolerances. It is not that easy to change the filters on software. First of all, it's not that easy to change anything on software. Secondly, most application software has not been designed to use any kind of filter, so there is nothing to check or change. So what do I mean by performing PM on our software and systems.

First, lets look at the way software breaks. In other words what kind of remedial care do we provide our systems. Then we can look at ways of preventing these breaks and prolonging the running life (UP-TIME) of our systems.

WHAT CAN GO WRONG?

Murphy's Law says that whatever can go wrong, will! The correlary to Murphy's Law is "Murphy was an optimist".

In the complex world of systems almost anything can go wrong. This can include:

HUMA N ERROR - This includes error at any
from level and it can be any area, whether it
oper is programming, ations or the user.

HARD WARE - This includes hardware-
huma induced errors or the n errors that arose
from when we tried to recover an actual hardware problem.

DISC SPACE - Who out there has never had
syst their program or em fail due to a lack
incl of disc space. This also udes those cases
beca where the program aborts use the al-
loted file size was not big enough.

PROG RAM ERROR - This can include a
this variety of things, but in instance, I am
assu talking about program mptions such as
inte "my program will never be rrupted".

WHAT IF IT'S NOT THE HARDWARE?

When something catastrophic happens (or even on minor interruption) it is almost always easier on the operations and programming personnel when the problem turns out to be hardware. After all, then "it's not my fault". You call the CE and wait until the problem is resolved. How many of us have been in a position where the machine went down because a room overheated due to an air-conditioning failure. Then not only do your operations personnel have someone else to blame, but so does your harware service representative.

On the other hand, if it turns out to be a specific operational problem --- such as the operator mounted the wrong tape or didn't follow the new procedures (that you just wrote neatly on the back of an envelope)--- then the operator suffers a tremendous ego loss. The same can occur to a programmer, although we adjust to that differently. You notice that we call any error a "BUG" --- as though some how it just crept in. We rarely say that there was an error on the part of the designer/developer/programmer. Instead we remark that we just "found a bug" as if we should be congratulated for our detective work instead of castigated for our failures in the development process. Ah well, the ego is a wonderful thing.

In any event, if we have an error it still will cause some loss of productive time on the part of the user, operations, programming, or all of them.

It is not my contention that we can eliminate all software errors though some magical process I call the PM. We don't solve all hardware problems though PM's. Besides the subject of producing better or error-free code is a whole separate topic that I have neither the interest nor inclination to address here.

Instead, let's focus on some simple steps to avoid the late night phone calls. After all, who wants to be interrupted from an amorous evening with your date by a phone call from the second shift operator telling you that the weekly MRP run blew off in the middle and he/she doesn't have a clue as to how to fix it. This leads me to my next thesis statement:

AN OUNCE OF PREVENTION IS WORTH A POUND OF LATE NIGHT PHONE CALLS

Actually there are many ways to avoid these calls. One of my customers simply leaves his phone off the hook at night. However there may be some other things that can be done to brighten up the picture. However, before I get into ways to fix things, I should point out that I do not advocate that you go back and apply these techniques to all existing jobstreams. After all, everyone has a million things that they would like to do to older systems but there is never the time or resources to accomplish that. Moreover, as Casey Stengel is reputed to have said --- "IF IT'S NOT BROKEN, DON'T FIX IT!"

BUILD MAINTAINABILITY INTO THE PRODUCT

Rather than try to develop procedures ex post facto, why not design them in at the start. Some examples of this include:

* CHECKPOINTS Consider designing some checkpoints into your programs and jobstreams. Checkpoints are places where the program/job identifies that it has completed a certain portion of the task and COULD be interrupted. This is very useful (and needed) on systems that will run a long time. Consider the frustration that occurs when a long program is running: Usually no one knows when it will be done but they don't wish to abort it because they will just have to start again. Say, for example, a DBLOAD. This is an example of a program from HP that probably should be checkpointed.

A good example from HP of this concept is the MM jobstreams. Many of these have checkpoints and a checkpoint/restart procedure to go with them.

* ASSUME THE WORST Remember Murphy. Programmers should not assume that everything is OK, especially if it would be very easy to make a verification check.

* PROGRAMMATICALLY CHECK FILE/DB CAPACITY Programmers should get in the habit of developing a simple routine to check the database capacity (use DBINFO) at the start of the program. Most of us have seen some application program that runs along until -- OOPS, the data base capacity is reached and the next record causes the program to abort. Usually there is little the operator/programmer can do except RESTORE the database to before the program ran and do some recovery. This involves expanding the database and then rerunning the job. It would be a lot simpler, if the program stopped before that problem

occurred and notified the operator to take appropriate action and then the program would pick up where it left off.

* HEURISTIC JCL There are examples of where the JCL learns what to do based on previous runs. For example, the MM/3000 MRP maintains a record of the size of the file the last few times the program was run. Based on that number and based on the transaction file, it establishes the file size requirements for all new files at run-time. This gets around the problem of creating a file that is too small as well as avoiding the problems created by always specifying a very large file and then only using a few records.

* GOOD DOCUMENTATION I shouldn't have to mention this, but the best way to make sure that your operators can solve problems is to provide quality documentation in a standardized format. Still, it always surprises me that many shops have little to no documentation.

* MANAGE YOUR DISC SPACE Sooner or later some program or system that you have will go "down the tubes" because you didn't have enough disc space. Moreover, it is really frustrating for programmers when they get blamed because the system ran out of disc space for a spooled report. Because this is such an important topic, I will get into this in more detail.

MANAGE YOUR DISC SPACE BEFORE IT MANAGES YOU

Like Zen, there are gurus of disc space management. Every one has his/her own unique mystical ways of addressing the problem. Usually, the problem arises in crisis. For example, it's a new month for the General Ledger and there is not enough room. All the players scurry around until someone astutely says "Let's purge Bill's account; he's on vacation this week."

Notwithstanding the shotgun approach, there are other ways to monitor and control your disc space. I would like to introduce some of the techniques of -

THE GENTLE ART OF DISC SPACE MANAGEMENT

LIMIT - Use the ;FILES=filespace parameter on the NEW/ALTACCT or NEW/ALTGROUP commands. If you are able to limit all groups and accounts so that the sum of all the limits is less than the total amount of disc space, you almost home free. Limiting still has its problems. First of all, you may be using a large amount of spool space, or there may be a lot of lost disc space (see LOST AND FOUND) so that you could still run out of available disc space. But the most obvious problem with limits is that sooner or later, some user will have his/her job aborted because they were the unlucky sole to run up against the limit. Because of this reason, I do not advocate the use of LIMITS except for accounts/groups which are not under central control and this is the only way you can keep these people in line. It is also good for the accounts you set up for users who want to "play" or use the system to do their homework for the programming course they are taking at the local night school.

MONITOR - A much more practical method is to begin to monitor the disc space so that you may take preventative action at an early stage. There are several ways to accomplish this. The quickest recap is to run FREE2 and monitor the amount of available space. By charting this data on a regular (at least once a week) basis you can get a good estimate as to when you are going to run into trouble. In conjunction with this, it is usually necessary to determine the answer to the question - "Who's using all the disc space?" or "Where the ** did all of our disc space go?" To determine who is using the disc space, you can use the REPORT command. This can be used in several ways.

1) REPORT of @.@ to a disc file and write a program to analyze the data and show you who has changed since the last report. (I think there is a contributed program that does this, but I can't remember which one. Also there are one or two third-party packages on the market that are supposedly capable of performing this monitoring function.) 2) REPORT to a hard copy device and analyze it by hand. Here's a simple trick to make that process easier. To get a quick recap at the account level just enter -

:REPORT ZZ@.@

This will get you a one line summary for each account as well as a breakdown for any groups that start with ZZ. Assuming that you have no groups that start with ZZ, you just get a summax for all the accounts. It is then reasonably easy to add these totals. Also if you have TDP you can use the TOTAL command as follows to have the system produce the total:

```
:RUN TDP.PUB.SYS
/:PURGE TOTALRPT
/:FILE TOTALRPT;SAVE;DEV=DISC;REC=-72,3,F,ASCII;NOCCTL
/:REPORT ZZ@.@,*TOTALRPT
/T TOTALRPT,UNN
/SET LEFT=13,RIGHT=22 << sets left/right to proper column >>
```

```
/TOTALQ 3/LAST.APPEND << ignore all the warning messages
about the line would be too long>>
/L LAST << last line now contains the total >>
```

LOST AND FOUND - (Or WHAT IS LOST DISC SPACE)

On a COOLSTART, you may have seen the prompt - RECOVER LOST DISC SPACE ? and wondered what it meant. When the system is running there are two file domains---system (permanent) and temporary. Temporary includes not only each user's temporary files but also the system spooled files. The system disc (or private volume) directory maintains entries only for permanent files. Temporary files for your session are maintained as part of the JDT (Job Directory Table) which resides in memory during the execution of your job/session. The system also maintains a separate table (IDD/ODD) for keeping track of spooled input and output files. As long as your session terminates and the spooled files are printed or deleted, you will not lose any file space.

When the system places a file on the disc, whether it is temporary or permanent, the disc's free space bit map is updated to reflect that the space is now used. When the file is purged, the free space map is also updated. However, if you have a system failure, OR if you RESTART THE SYSTEM (with other than a WARMSTART) and you had spooled files to be printed you will have lost disc space. In other words, the disc free space bit map indicates that the space is in use, but the system has no idea of what that space is used for and no way to get to it. It is therefore "LOST DISC SPACE". The only way to reclaim this is to answer YES to RECOVER LOST DISC SPACE? on a COOLSTART, or do a complete RELOAD. Depending on how long it has been since your last reload (or last RECOVERy of LOST DISC SPACE) this can amount to many sectors of lost space.

One way to tell how much lost disc space you have is to use the following formula:

LOST = TOTALAVAIL - (FREESPACE + INUSE + OVHD) where

TOTALAVAIL = Amount of usable formatted disc space based on the number and type of disc drives you have.

Use the following numbers:

DRIVE TYPE	# AVAIL SECTORS	FORMATTED CAP IN BYTES	LOST SECTOR DEFECTIVE *
7906	76,800	19,660,800	48
7911	109,824	28,114,944	0 (64)*
7912	256,256	65,601,536	0 (64)*
7914	516,096	132,120,576	0 (64)*
7920	195,600	50,073,600	48
7925	469,440	120,176,640	64
7933	1,579,916	404,458,496	0 (92)*

* CS80 drives have built in sparing for the first defective track. Otherwise use the number in parenthesis. This column shows how many sectors will be lost if you delete a specific track. If a track is reassigned, you do not lose any capacity---only the time spent for extra head movement.

FREESPACE = Total available as shown by FREE2.

INUSE = Sum of DISC space used as shown by REPORT.

OVHD = Constant for your specific configuration. This number allows for the system directory, virtual memory space, system disc tables, disc labels and free space maps. You have to compute this number after a RELOAD or COOLSTART (with RECOVER LOST DISC SPACE) so that you will be sure that LOST = 0.

Once you have computed the OVHD number it will remain constant as long as you don't change your configuration. The addition of a few IO devices will not affect this number significantly, but a change in the size of your directory or in the amount of virtual memory will have an impact.

If you now monitor your disc space, you can always make a quick check to determine if you have any lost disc space.

In addition to RECOVER LOST DISC SPACE and complete RELOAD, you can regain space on a private volume with the VINIT subsystem and the > COND ldn;RECOVER command.

OUT WITH THE OLD

The last option (besides buying more discs from your friendly HP sales representative) is to get rid of some of the unused files. Unfortunately it is usually hard to get a consensus as to what is "unused". There are several techniques to determine which files have not been accessed since a certain date. These include the :STORE command with the ;DATE <= parameter, and several utilities (LISTFXX from your friendly S.E. or MPEX from your friendly independent third-party vendor). These allow you to identify "old" files.

I recommend to my customers that they get in the habit of storing to tape all files which have not been used in 6 months and purging these from the system. Somehow everyone understands

the process, but no one seems to implement it on a regular basis. Oh well, I'm not really hurt--- I guess they just keep on buying ! more discs.

This brings us to Covitt's law of disc space which states:

"The amount of disc space used rapidly climbs until it reaches the total amount of usable disc space... AND STAYS THERE FOREVER. (Until more discs are added. Then it rapidly climbs until it reaches the total amount of usable disc space and stays THERE forever. Until ....)"

MAKE THE MOST OF WHAT YOU'VE GOT

Assuming that you cannot add more disc space (due to lack of budget or other critical resource), your best option is to try and use less disc space to hold your existing data. Assuming you have already tried to remove the old files. REMEMBER - (OLD FILES NEVER DIE --- THEY JUST TAKE UP DEAD DISC SPACE)

There are several ways to reduce the amount of space used by your existing files. However some of the methods have limitations or other operational drawbacks. But here's a sampling of things you can try:

REBLOCK existing files.
This allows you to continue your applications without change and may return some disc space. LISTFXX gives you an indication of how much space you could reclaim if you reblocked. However, this may not make a lot of sense if you have a job stream that recreates the file with the old block factors.

PURGE those dead files.
This has already been mentioned, but I want to emphasize that are some other files (like EDITOR/TDP K - files) that can be purged without even worrying about backing them up.

SCRUNCH files.
This is a utility that moves the LIMIT on the file back to the current EOF. It can be very helpful on files that change in size from run to run. Create the files with enough records (;DISC= nnnn,32,32) and then run SCRUNCH on the file immediately after it is closed. If you do your own FCLOSE on the file, you can do the same thing by making sure that the DISP parameter is set to return space beyond the EOF. However, this does not work well with KSAM files and any file that is to be later used with APPEND access.

COMPRESS files.
This utility stores the data in a compressed format, usually by doing blank compression. There is a corresponding utility to UNPRESS the files. This will usually result in a significant disc space saving, but will require that the file be expanded before it is accessed by almost any program. You might do just as well to ARCHIVE the file to tape.

Along with COMPRESS, you can save space by saving files in the VARIABLE format instead of FIXED format within the EDITOR. This will save a little space for JCL files by simply truncating the trailing blanks.

ARCHIVE files.
As mentioned above you can archive files to tape. Several utilities exist in the contributed library to manage this feature. These will typically be data files that won't be accessed for a while. Good candidates for this are the history files ---like the last seven years of sales data that marketing wants to keep ONLINE.

LIMIT YOUR DATABASE CAPACITY

Because the IMAGE database manager reserves all your allowable space immediately, you may have some wasted or at least available space within your databases. Because of the way that the space is originally allocated, the MPE file system thinks that the LIMIT and EOF are the same and that 100% of the space is in use. However, if you look within the database, you will often find that only a small portion of a particular dataset is actually in use.

It is not an easy task, however, to constantly play with the database capacities. If you have no outside utilities, you would be forced into a complete DBUNLOAD/DBLOAD. I would recommend that you waste disc space rather than your time in this case. However if you happen to have ADAGER or some of the utilities in the contributed library, you might try to allocate your disc space more conservatively and assume that you will be able to increase it if you need to. However, that may create some other problems down the road. This requires that you monitor your data base capacity and brings us to the next topic:

WATCH YOUR DATABASE CAPACITY

Even if you are not taking conservative approaches to allocate your data base capacity, sooner or later you will have to increase it. The question is not "IF" but "WHEN". In order to monitor this you can periodically run QUERY and do a FORM SETS on each database, or have a program perform this task. I have written a simple COBOL program (and contributed it to the library) called DBCAPCHK.

DBCAPCHK simply calls DBINFO for each database that you pass to it. It provides a simple report that shows percentage full for each dataset. It can easily be enhanced to start tracking some statistical data and develop trend analysis. Those modifications are left to the reader as an exercise. (I always hated that caveat in the math textbooks, but I thought I'd put it in here anyway!)

A RETURN TO SOFTWARE P.M.'s

Having digressed for a while onto the subject of disc space management, let's return to the generic subject of Preventative Maintenance for software. Actually watching your disc space is probably the single most important aspect of software PM's.

I would like to suggest to the reader that you implement some periodic jobstreams to perform the PM. These should be at least on a weekly basis. I would like you to establish a Job that includes

1) FREE2
2) REPORT ZZ@.@
3) MEMLOGAN
4) LOGERR (or IOERR) (formerly LOGUTIL)
5) DBCAPCHK
6) LISTFXX
7) PURGEK (JANITOR or something else that PURGES K-Files)

Recently HP has been introducing to most customers the TELESUP program and associated files. Many of the old-time "SE contributed" programs that were "unsupported" but essential have now been identified as supported utilities through the TELESUP program. Also included with these is a jobstream that includes FREE2, MEMLOGAN, TUNER5, and IOERR. The hardware folks have found that keeping a regular watch on your system will alert them to any significant problems.

My hope is that those of us concerned with the software side will extend this philosophy and also watch disc space and data base capacities. Also I'd like to reiterate my request that you look at archiving old files on a monthly basis.

SUMMARY

Yes, we've finally come to the end. I hope that the reader will take the thoughts and ideas included in this paper and develop a regular preventative jobstream for his/her shop. The long-term rewards from preventing major disasters and eliminating the late night phone calls will more than compensate for the start-up effort. So start doing your software PM's NOW!

For additional material on this topic, I suggest you read the article in these proceedings on the topic of managing your system written by my fellow S.E., John Vega.

BIOGRAPHY FOR

Marc Covitt Systems Engineer Hewlett-Packard 9606 Aero Drive San Diego, CA 92123 (619) 279-3200

Marc has been with Hewlett-Packard for 9 1/2 years in various positions. He has been a Systems Engineer for 2 1/2 years. Marc holds a bachelors degree from M.I.T. and a Masters in Computer Science from West Coast University. He was awarded the C.D.P. certificate in 1980.

He has been a regular contributor at International, Regional and Local User Group Meetings. He has had articles published in the HP User's Group Journal. Marc is an accomplished speaker and gives great talks.

In 1980, he was named by the International Users Group as the outstanding contributor to the Users Group within Hewlett-Packard.

Performance Programs and The Measurement Interface

Bryan Carroll
MA-COM

There are times when each of us needs the capabilities of one or more of the performance programs. Programs such as OPT and SOO are used to monitor the performance of individual programs or of the entire system and are widely used in HP3000 environments around the world. These performance programs and others like them are often misunderstood, and, in some cases, information derived from the programs can be inaccurate. This paper will attempt to interpret the data from these programs starting with the MPE Measurement Interface.

The Measurement Interface is a data gathering facility of the HP3000 that has evolved from one operating system release to the next. Its immediate predecessor, MMSTAT, could collect much of the same data but did not make the data available to user programs in a usable format. The only procedure provided by MMSTAT required a dedicated tape drive and a great deal of time to produce meaningful reports. The Measurement Interface that has been released with MPE IV has made data, in the form of event counts, available to priviliged mode programs. MMSTAT had provided a trace of events, each of which could be distinguished from the preceeding and succeeding events in time. Now, with the Measurement Interface, any time dependent relationships must be determined programmatically.

In order to ensure that good data is collected from the system, some design goals must be established. These goals must include a minimum of system resource drain and easy access by a performance program. HP seems to have provided both of these goals in their design of the Measurement Interface as well as a sound base from which to expand. This base takes the form of a system extra data segment which holds the measurement data. This method of implementation could provide a stable base which would not have to change from one operating system release to the next.

The Measurement Interface control information is kept in a system extra data segment that is created and initialized by INITIAL when the system is booted. The data segment is not required to be in bank 0 but is locked and frozen to ensure that it is always in core. The locked and frozen requirement allows the system to access it much more quickly causing much less system overhead. The access time is similar to a load or store instruction since there will never be a segment fault. (A segment fault is caused by accessing an extra data segment that is not in core but out on disc in virtual memory)

The Measurement Interface makes use of some memory locations, other than those in its extra data segments. The Interrupt Control Stack (ICS) has three words reserved for the Measurement Interface, one word for flags and the other two for holding a pause time. Four words in the System Global (SysGlob) area of bank 0 are used to hold some flags and the absolute memory address (bank and offset) of the control extra data segment. The Process Control Block Extention (PCBX) for each process reserves four words to hold times and flags for that processes activity. Each of these places in memory is used to compute pause times or counts to later be stored in the appropriate Measurement Interface extra data segment.

I. Three Types of Statistics Gathered

There are presently three different types of information that can be enabled and disabled selectively or as a group. These three are Global, Input/Output, and Process related statistics. Each type of statistic is kept in a separate extra data segment which is also locked and frozen in place to speed access. The control extra data segment, established by INITIAL, contains pointers to all other extra data segments and is also used to hold the Global statistics when they are enabled. A counter is kept for each type of statistic to determine when to build an extra data segment and when one can be discarded. The counter is incremented each time the Measurement Interface is initiated by a program, such as OPT, and decremented each time the same type of program terminates one or more types of statistic gathering. This will prevent one program from deleting a Measurement Interface extra data segment while another program is still using it. The control extra data segment, also called the Measurement Information Table, contains pointers and work space for the shared clock Interface and space is also reserved for a future HP performance program called TRACER.

Global statistics are futher subdivided into classes, although only one class is currently defined. Each class is then subdivided into subclasses which are divided into groups.

Subclass 0 of the Global statistics is used to hold the counters for system wide CPU pauses, swaps and other dispatcher/memory manager activities. These numbers can be very helpful in determining the overall performance of your machine. These numbers can indicate possible problems such as thrashing, CPU overload, Disc bottle necks and similar problems. All other statistics are limited to specific processes or pieces of hardware.

Subclass 1 Global statistics focus on disc activity. There is one group entry for each disc configured on your system. There are counters for blocked and unblocked reads and writes as well as for memory manager reads and writes. All of the numbers collected in this subclass are in terms of physical disc I/O and no information can be derived about logical I/O's.

Lineprinter and Magnetic Tape activity consume the last two subclasses, subclass 2 and 3 respectively. There is one group in each of the subclasses for each lineprinter or tape drive that is configured into the system. Within each group there are three counters. The numbers of device reads, device writes and control operations are kept for each device for consistency. The counter for lineprinter reads is not used but exists for uniformity between devices.

The Input/Output statistics are kept as class 14 statistics. These statistics include information about all types of I/O. The groups are each 16 words long and contain copies of either the Input/Output Que (IOQ) entry or Disc Request Que (DRQ) entry for disc I/O. The device drivers add and delete entries to this table directly. This type of statistic is very dynamic and would require some analysis before it could be presented in a usable format. Due to the dynamic nature of this type of statistic, the table would have to be frozen while it was either analyzed or copied to another place for analysis. A sampling interval becomes very important during analysis of this statistic and would require some study of the machines hardware and some experimentation to choose correctly. Choosing a sampling interval too long would allow entries to appear on the list and then be removed before they were analyzed. A sampling interval chosen too short may not allow enough activity to happen on the system for proper analysis. The speed of all active I/O devices should be considered in determining a sampling interval.

Class 15 statistics are known as process statistics. This group of statistics can be viewed as a table indexed by the process identification number (PIN) of each active process. The first entry, entry 0, is an overhead entry and contains global information about the operating system release level and the time sampling began. Each entry contains 52 words of process specific information including processing times and some I/O counts. It is important to note here that the counters are initialized to zeros when the extra data segment is created, or when the entry is added to the table. If a program had accumulated some CPU time or disc I/O's prior to enabling process statistics, these times and counters will not be included in the figures presented by the Measurement Interface. It is also important to note that your program that uses the Measurement Interface may not have been the program that actually initiated the statistics gathering. The statistics gathering begins when the first program that requires the Measurement Interface requests it to start and will not terminate until ALL processes using the statistics have disabled access to the Measurement Interface. All the times and counters in the process statistics relate only to a single process and include counters, CPU time, disc I/O's, pause times for terminal reads and disc I/O.

II. Access Routines for The Measurement Interface

There are five (5) intrinsics defined by Hewlett Packard for use with the Measurement Interface. One of these routines is not yet implemented and another routine, although written and included in the MPE Kernels, is not used. The routines as they stand now are not complete enough to use without a user written retrieval routine. The five routines are: STARTSTATISTICS, STOPSTATISTICS, UPDATESTATISTICS, GETSTATISTICS and GETPROCSTATS.

The STARTSTATISTICS routine is used to enable one or more classes of statistics for gathering. The appropriate bits are set in the System Globa y area to communicate to MPE which classes of statistics are enabled for data gathering. If a class of statistic is being enabled that had not previously been enabled, the necessary system extra data segment(s) is built, frozen and locked in place. The Data Segment Table number (DST), the bank number and the offset are put in the Measurement Interface control extra data segment for future references. The enabled counter for each class of statistic being enabled is incremented. If a new data segment is created for a class of statistic, it is initialized to the appropriate values. Starting statistics on a heavily loaded system (one with many active processes) could require a significant amount of overhead. During initialization of process statistics, the initialization routine must obtain a program name, session or job number and the current state of each process. This procedure could cause many disc accesses. Continual starting and stoping of statistics gathering at the process level (class 15) should be minimized because of this overhead.

STARTSTATISTICS is an integer procedure with one parameter. The routine will return one of four possible values indicating success or failure. A value is not returned to indicate that statistics gathering has been active prior to this call to STARTSTATISTICS. You will be unable to determine from this routine whether or not statistics have just begun or if statistics were being gathered prior to your call to STARTSTATISTICS. The one parameter is a bit mask indicating which class or classes of statistics to start.

STOPSTATISTICS will disable one or more classes of statistics gathering. The class enabled counter is decremented for the requested class. If the counter is then zero, meaning there are zero remaining accessors to this class of the Measurement Interface, the corresponding system extra data segment will be deleted. The bit in the System Global (SysGlob) area corresponding to the class of statistics being disabled will be reset to reflect the termination of statistics gathering for this class. If STOPSTATISTICS is not called prior to process termination, (ie. if the program aborts or the programmer forgets to code a STOPSTATISTICS!) the system will call STOPSTATISTICS for the process. This should prevent the Measurement Interface from being active while no process has it enabled, ensuring a minimum of system overhead.

This routine, like STARTSTATISTICS, has only one parameter but it is not an integer procedure. There is not a return value so the condition code should be checked to determine the success or failure of the routine. The parameter is a bit mask that corresponds to the bit mask used for STARTSTATISTICS.

There is a System Internal Resource number (SIR) reserved for the Measurement Interface. This SIR should not be locked by any user programs. Its use is reserved for the START and STOPSTATISTICS routines only.

The UPDATESTATISTICS routine is written and exists in the KernelC segment of MPE. This routine was written to update the statistics in a given class, subclass and group but is currently not used by MPE. The statistics are currently updated only by inline code throughout the operating system. There is a secondary entry point to this routine that will bypass all parameter checking. This entry point will execute faster bacause of the elimination of paramenter checking but is still much slower than inline code. The inline code is faster but Hewlett Packard will pay the price of speed in supportability. Any changes to the Measurement Interface will require extensive changes to the MPE Kernels updating the data. This routine should never be called by a user program. If a user program was allowed to update the statistics, the data would become meaningless.

Access to class 0 statistics (Global Statistics) is accomplished through the GETSTATISTICS routine. This routine will do parameter checking and validity checking of the class and subclass. There is an entry point, FGETSTATISTICS, that will excute faster by skipping the parameter checking but, it is more likely to cause errors (because of not checking parameters.) This entry point should only be used after a program has been debugged and you are satisfied that your program is working properly. The request for statistics may be as small as one word or as large as the entire table. This is the only routine provided by Hewlett Packard to retrieve data from the Measurement Interface and will only work on class 0 statistics.

The GETPROCSTATS procedure has not yet been implemented. It is planned for this routine to access the process (class 15) and I/O (class 14) statistics in a manner similar to the GETSTATISTICS routine. A secondary entry point, FGETPROCSTATS, is planned to speed execution by eliminating parameter checking. A substitute routine would not be difficult to write and I will provide one upon request.

All five (5) routines used to access the Measurement Interface programmatically require privileged mode to execute and are fairly safe routines to use.

III. Inline Code Updates The Measurement Interface

Currently, all updating of the Measurement Interface data is done through inline code in the operating system. This code appears in many places throughout the MPE source code including KernelC, KernelD, Hardres and the device drivers. Figure 1 shows some examples of Q-MIT KernelC inline code used to update the Measurement Interface statistics.

```
PAGE 0270   KERNELC          DISPATCHER : DSP

21140000  01720 3      END;                                                    <<04485>>
21142000  01721 2
21144000  01721 2   ASMB(ZERO,DZRO);
21146000  01722 2   TOS.DISPRUNNINGFLAG:=1;
21148000  01723 2   DISPTOAWAKEMSG:=TOS; <<===>DISPATCHER RUNNING,NOT PAUSED>>
21150000  01724 2   ABSOLUTE(CPCB):=TOS;
21152000  01726 2   TRL(4):=TOS; <<SET QTIME TO 0 - DON'T WANT CLOCK INTERRUPTING>>
21154000  01727 2   ENABLE;
21156000  01730 2
21158000  01730 2
21160000  01730 2   <<
21162000  01730 2   WHO WAS RUNNING LAST?
21164000  01730 2   >>
21166000  01730 2
21168000  01730 2   ASMB(TEST);
21170000  01731 2   IF <> THEN
21172000  01732 2      BEGIN <<A PROCESS WAS RUNNING>>
21174000  01732 3      IF S0=-1 THEN
21176000  01735 3         BEGIN  <<SYSTEM JUST COMING UP>>
21178000  01735 4         ASMB(DEL);
21180000  01736 4         INITIO(2); <<INITIALIZE SYSTEM DISC>>
21182000  01740 4         STARTCLOCK(0,0D); <<GET CLOCK MOVING>>                  <<01770>>
21184000  01742 4         END
21186000  01742 3      ELSE
21188000  01745 3         BEGIN
21190000  01745 4         LASTPROCINX:=TOS-SYSBASE;
21192000  01750 4         LASTSTKSYSBASEINX:=ICS'STKOST&LSL(2)+DSTSYSBASEINX;
21194000  01754 4         IF GCLASSENABLEDMASK.CLASS0 THEN
21196000  01761 4            BEGIN  <<MEASURE PROCESS BURST EVENT AND DURATION>>
21198000  01761 5            TOS:=MEASSTATXDSBANK;
21200000  01764 5            TOS:=MEASSTATXDSBASE;
21202000  01767 5            TOS:=TOS+COSUB0'SEGRELOFF+C'LAUNCH;                 <<RAY.V>>
21204000  01772 5            ASMB(LSEA);
21206000  01773 5            TOS:=TOS+1;
21208000  01774 5            ASMB(SSEA);  << CUM # OF LAUNCHES>>
21210000  01775 5            TOS:=TOS-C'LAUNCH+C'CPUPROCESS;                     <<RAY.V>>
21212000  01777 5            ASMB(LDEA);
21214000  02000 5            ASMB(ZERO;RCLK);
21216000  02002 5            ASMB(DADD);
21218000  02003 5            ASMB(SDEA;DDEL); <<CUM CPU TIME ON PROCESSES>>
21220000  02005 5            END;
21222000  02005 4         IF GCLASSENABLEDMASK.CLASS15 THEN                      <<01812>>
21224000  02012 4            BEGIN <<CPU TIME & NUMBER OF LAUNCHES>>             <<01812>>
21226000  02012 5            TOS:=MEASPROCXDSBANK;                               <<01812>>
21228000  02015 5            TOS:=MEASPROCXDSBASE;                               <<01812>>
21230000  02020 5            TOS:=TOS+((LASTPROCINX+SYSBASE-ABS(PCBP))/PCBSIZE)* <<01812>>
21232000  02026 5                 CLASS15'SUBOSIZE+CP'LAUNCH;                    <<01812>>
21234000  02031 5            ASMB(LSEA);                                         <<01812>>
21236000  02032 5            TOS:=TOS+1;                                         <<01812>>
21238000  02033 5            ASMB(SSEA);                                         <<01812>>
21240000  02034 5            TOS:=TOS-CP'LAUNCH+CP'CPUTIME;                      <<01812>>
21242000  02036 5            ASMB(LDEA);                                         <<01812>>
21244000  02037 5            ASMB(ZERO;RCLK);                                    <<01812>>
21246000  02041 5            ASMB(DADD;SDEA;DDEL);                               <<01812>>
21248000  02044 5            END;                                                <<01812>>
21250000  02044 4         END;
21252000  02044 3      END
```

Figure 1

IV. Performance Programs

The Measurement Interface is only one step in a three step procedure to obtain an accurate picture of the performance of any machine. These three steps are the collecting of data (measurement Interface), the logging of the data and the presentation of this data in an understandable format (performance program).

The logging of the data is performed internally by each performance program. Large disc files or tape files are not used to log the data collected by these performance programs. For our discussion, the logging of data will be included in the discussion of the means of presenting the data.

The performance programs available on the HP3000 are mostly contributed programs whose authors are unknown, whose source is 'unavailable', or for some other reason, are unsupported. These programs are needed and until OPT (On Line Performance Tool) was released, they were the only user runable programs available for system performance monitoring. I was able to find eight (8) separate performance programs for comparison and discussion. I have chosen this group of programs to represent both the most used performance programs and the largest variety of programs. The programs I have chosen are: SOO - four different versions, MOO - a takeoff from SOO, OPT - HP's On Line Performance Tool, Surveyor - another performance program, and Porpoise - a Boeing library program. The source code to OPT was not made available at the time of this writting and will be excluded from some of the discussions.

Each program periodically updates a process display screen, except for Porpoise which displays a single line of CPU statistics. The purpose of this part of the discussion is to compare and contrast the process displays of the various programs from a viewpoint of reliability and accuracy. Some of these programs will use the Measurement Interface and others will not. Each program will use system level routines at different levels. It is hoped that this discussion will give you the information needed to determine the best performance program or programs to meet your needs. I do not intend to critique or recommend performance programs, but to give you the information necessary to allow you to examine them for yourself.

All of the selected programs use privileged mode at some point in their processing. Surveyor runs in privileged mode for the entire span of its run while all the others will execute privileged mode instructions as they need it. A few programs try to improve their response times as well as to provide more timely data by raising their own priority. This is sometimes necessary to compete with high priority system processes or more likely high priority applilcations. MM3000, for example, runs its monitor at priority 100 or higher in the AS que. Table 1 lists the programs which use these intrinsics (GETPRIVMODE or GETPRIORITY). The programs which do not list GETPRIVMODE as an intrinsic, gain privileged mode through a user written procedure.

All programs except for one version of SOO use various MPE undocumented routines. These routines are written for the operating system to use and, for one reason or another, were not released as supported MPE intrinsics. Many of these routines require privileged mode. Documentation for some or all of these routines may be obtained from a 'helpful' HP SE or by attending some system level courses taught by HP SE's. A brief statement about each of these routines may be found in the Appendix. Consult Table 1 for those programs which use undocumented routines.

Program Name	Initial Stack	Max Running Stack	Dangerous Intrinsics	System (Undocumented) Routines
1) SOO (1)	7878	9508	GETPRIVMODE	none
2) SOO (2)	4549	8744	GETPRIVMODE	ATTACHIO
3) SOO (3)	12135	13764	GETPRIVMODE	DMOVE EXTIN' INEXT'
4) SOO (4)			GETPRIORITY	ATTACHIO CHECKLDEV GENMSG GET'DSDEVICE GET'DSXREF

					PROCFILE RESETDB SETSYSDB
5)	MOO	2121	7480	GETPRIORITY GETPRIVMODE	ATTACHIO CHECKDISC CHECKLDEV GENMSG GET'DSDEVICE GET'PAGE GET'SIR LOCK'DFS'DATA'S RESETDB SCAN'PAGE SETCRITICAL STARTSTATISTICS STOPSTATISTICS SETSYSDB UNLOCK'DFS'DATA
6)	OPT	3639	9660	GETPRIVMODE	ATTACHIO FGETSTATISTICS FINDDEVICES GENMSGU INEXT' PROCFILE STARTSTATISTICS STOPSTATISTICS THISCPU

Program Name	Initial Stack	Max Running Stack	Dangerous Intrinsics	System (Undocumented) Routines
7) SURVEYOR	1068	8840	GETPRIORITY	DELAY FINDDEVICES GETSTATISTICS STARTSTATISTICS
8) PORPOISE	1131	3827	GETPRIVMODE	GETSTATISTICS STARTSTATISTICS STOPSTATISTICS

Table 1

Figures 2 through 8 show a sample of the process display of each of the performance programs. The first column appearing on most displays is cumulative CPU time. All programs except Surveyor display this statistic. This number is kept in two places in the system, but all of the programs obtain this number from the same location, the PCBX area of the processes stack. This place seems to be the most reliable one since this location is used to update the numbers found in the "Report" command upon process termination. A copy of this number is also put in the logfiles to be used for accounting or billing purposes. There is no reason to suspect any inaccuracy in the other location that supplies this number. This other location is in the Measurement Interface process statistics. The first location where cumulative CPU may be found, in the PCBX area of the stack, is a better choice for this type of display because it supplies a cumulative CPU time since the process began. The Measurement Interface can only provide the number of CPU seconds that have been used since the Measurement Interface has begun.

```
        SON OF OVERLORD          VERSION IV
FILE                        USER                CPU                    STACK
NAME                        NAME                TIME   %  Q  J/S#    SIZE  PIN PRI
------------------------- ----------------- ------ -- - ------ ----- --- ---

C.I.                        TEST     .GNET        5  0 C S25     4920  14 152
C.I.                        MGR      .PAYROLL     2  0 C S26     6008  15 152
QUERY    .PUB      .SYS     MGR      .PAYROLL     2  0 C S27    16732  32 152
TEST     .PUB      .PAYROLL MGR      .PAYROLL    17  2 C S6      8340  33 154
C.I.                        ACTUAL   .EPS         3  0 C S125    2696  53 152
C.I.                        CONSOLE  .OPERATOR    1  0 C S103    4744  58 152
XXFCS    .EARLE    .EPS     GENERAL  .EPS       252  0 C S16    25852  59 152
QUIC302  .PUB      .QUASAR  CASH     .EPS        10  0 C S138   10896  61 152
XXFCS    .EARLE    .EPS     GROUPX   .EPS       214  0 C S73    25852  63 152
COBOL    .PUB      .SYS     MGR      .DESIGN     24 19 c S140   29940  70 200
MBQ      .UTIL     .SYS     MANAGER  .SYS        26  0 D J19     5904  87 202
XXFCS    .EARLE    .EPS     GROUPM   .EPS       936  0 C S119   26620  98 152
XXFCS    .EARLE    .EPS     GROUPX   .EPS       960  0 C S45    25724 101 152
SOO1     .TOOLS    .TECH    BRYAN    .TECH        1  2 C S43     8796 114 1
RELATE   .PUB      .CRI     ACTUAL   .EPS        50  0 C S92    19492 119 152
TP3000   .PUB      .CCC     PSR      .PAYROLL    20  1 C S131    8916 131 152
C.I.                        AWAYNE   .LABOR       4  0 C S129    5336 137 152
MAILROOM.HPMAIL    .SYS     MAILROOM .HPOFFICE    1  0 D J41     5212 142 202
C.I.                        MGR      .PAYROLL     1  0 C S7     4752 153 152
TEST     .PUB      .PAYROLL MGR      .PAYROLL   219  1 C S11     8340 167 152
TEST     .PUB      .PAYROLL MGR      .PAYROLL    80  2 C S9      8340 173 152
XXFCS    .EARLE    .EPS     SALES    .EPS       401  0 C S80    25596 175 152
TP3000   .PUB      .CCC     ENTRY    .PERSONNL    1  0 C S133    7380 182 152
XXFCS    .EARLE    .EPS     MACOM    .EPS       170  0 C S100   25724 187 152

TIME USED:   4.726 CPU SEC;  17.189 ELAPSED SEC.  27.494% UTILIZATION.

CHANGES TO STATUS LIST

C.I.                        MGR      .PAYROLL    ADDED.
TP3000   .PUB      .CCC     MGR      .PAYROLL    DELETED.
```

Figure 2 SOO version 1

```
        SON OF OVERLORD          VERSION 1D
FILE                        USER                CPU                    STACK
NAME                        NAME                TIME   %  Q  J/S#    SIZE  PIN#
------------------------- ----------------- ------ -- - ------ ----- ----

C.I.                        TEST     .GNET        5  0 E S25     4920   14
C.I.                        ACTUAL   .EPS         3  0 L S125    2696   53
C.I.                        CONSOLE  .OPERATOR    1  0 L S103    4744   58

TIME USED:   0.000 CPU SEC;   2.253 ELAPSED SEC.   0.000% UTILIZATION.
```

Figure 3 SOO version 2

```
 ** SON OF OVERLORD ** Version 1G **   MPE IV  C.C1.A0.    (  30 Sec. delay )
File name                  User name          CPU     CPU %    J/S#   Stack
@.@.@                      @.@                time      .00 Q @      size  Pin
-------------------------- ------------------ ------ ------ - ------ ----- ---


 CI(RUN MIS030.PUB        ) TEST     .GNET          5    .00 C S25     4920  14
 CI(RUN QUERY.PUB.SYS     ) MGR      .PAYROLL       2    .00 C S26     5200  15
 QUERY.PUB.SYS              MGR      .PAYROLL       2    .00 C S27    16732  32
 TEST.PUB.PAYROLL           MGR      .PAYROLL      17    .00 C S6      8212  33
 CI(SHOWDEV BANKMUX2      ) ACTUAL   .EPS            3    .00 C S125    2696  53
 CI(RUN SPOOK.PUB.SYS     ) CONSOLE  .OPERATOR       1    .00 C S103    4744  58
 XXFCS.EARLE.EPS            GENERAL  .EPS          252    .00 C S16    25852  59
 XXFCS.EARLE.EPS            GROUPX   .EPS          214    .00 C S73    25852  63
 COBOL.PUB.SYS              MGR      .DESIGN       43  10.27 C S140   28916  70
 QUIZ104.PUB.QUASAR         AWAYNE   .LABOR         1    .08 C S129    4608  86
 MBQ.UTIL.SYS               MANAGER  .SYS           26    .00 D J19     5904  87
 XXFCS.EARLE.EPS            GROUPM   .EPS          996  30.64 C S119   26620  98
 XXFCS.EARLE.EPS            GROUPX   .EPS          985  37.03 C S45    25724 101
 S003.TOOLS.TECH            BRYAN    .TECH           1   2.19 C S43    13052  11
 RELATE.PUB.CRI             ACTUAL   .EPS           50    .00 C S92    19492 119
 TP3000.PUB.CCC             PSR      .PAYROLL       21    .00 C S131    8916 131
 MAILROOM.HPMAIL.SYS        MAILROOM .HPOFFICE      1    .00 D J41     5212 142
 CI(RUN TEST              ) MGR      .PAYROLL       1    .00 C S7      4752 153
 TEST.PUB.PAYROLL           MGR      .PAYROLL     220    .00 C S11     8212 167
 TEST.PUB.PAYROLL           MGR      .PAYROLL      82   2.58 C S9      8340 173
 XXFCS.EARLE.EPS            SALES    .EPS         401    .00 C S80    25596 175
 TP3000.PUB.CCC             ENTRY    .PERSONNL      1    .00 C S133    7380 182
 XXFCS.EARLE.EPS            MACOM    .EPS         170    .00 C S100   25724 187
<<  23 Displayed                                                      312656  >>
<<   0 Not displayed                                                      0  >>
<<  23 Total active PCBs                                              312656  >>

 Time used:  11.274 CPU sec.;  13.638 Elapsed sec.  82.666% Utilization.

 Changes:     0 Added     0 Deleted                (  4:18 PM)
```

Figure 4 S00 version 3

```
SON OF OVERLORD IV   VERSION BG.15/SS.V1          MON, NOV 14, 1983,  4:19
U MODE   DELAY = 60                               HIGHLITE= BRYAN    .TECH
:COMMAND OR PROGRAM        USER NAME          CPU  % Q Wc J/S# STACK   XDS   C
:RUN MIS030.PUB            TEST     .GNET       5    C Bt S25   4920 31568     0
:RUN QUERY.PUB.SYS         MGR      .PAYROLL    2    C Bt S26   5200  3348     0
QUERY    .PUB      .SYS    MGR      .PAYROLL    2    C Bt S27  16732     0 28172
TEST     .PUB      .PAYROLL MGR      .PAYROLL   17    C Bt S6    8212  2380     0
S004    ?TOOLS     .TECH   BRYAN    .TECH       1    C a  S43   8044  5992     3
:SHOWDEV BANKMUX2          ACTUAL   .EPS        3    C Bt S125  2696 18912  5532
:RUN SPOOK.PUB.SYS         CONSOLE  .OPERATOR   1    C Bt S103  4744  6068     0
XXFCS    .EARLE    .EPS    GENERAL  .EPS      252    C Bt S16  25852  1528     0
XXFCS    .EARLE    .EPS    GROUPX   .EPS      214    C Bt S73  25852  6220     0
MBQ      .UTIL     .SYS    MANAGER  .SYS       26    D T  J19   5904  3252 12892
XXFCS    .EARLE    .EPS    GROUPM   .EPS      996    C Bt S119 26620 15208     0
XXFCS    .EARLE    .EPS    GROUPX   .EPS     1015    C A  S45  25724  9388     0
RELATE   .PUB      .CRI    ACTUAL   .EPS       50    C Bt S92  19492  8452     0
QUIZ104  .PUB      .QUASAR AWAYNE   .LABOR      0    C Bt S129  4304     0  3124
TP3000   .PUB      .CCC    ENTRY    .PERSONNL   0    C bs S133  5332  1948     0
TP3000   .PUB      .CCC    PSR      .PAYROLL   21    C Bt S131  8916 15972 12520
MAILROOM.HPMAIL    .SYS    MAILROOM .HPOFFICE   1    D Q  J41   5212  1864  7452
TP3000   .PUB      .CCC    MGR      .PAYROLL    0    C bs S7    5140 12180  7812
TEST     .PUB      .PAYROLL MGR      .PAYROLL  220    C Bt S11   8212 13732 23976
TEST     .PUB      .PAYROLL MGR      .PAYROLL   84    C Bt S9    8340  2380     0
XXFCS    .EARLE    .EPS    SALES    .EPS      401    C Bt S80  25596 10552     0
```

```
:COBOL PFLSR021.FLSSRCE,U  MGR       .DESIGN      1    C Bt S140   5296   8492      0
XXFCS    .EARLE    .EPS    MACOM     .EPS       170    C Bt S100  25724  31212      0
MIS030   .PUB      .GNET   TEST      .GNET       10    C A  S25   30796    588      0

/SOO:e
```

Figure 5 SOO version 4

```
MOO  (Mistress of Overlord)        Version  ( RW ) MON, NOV 14, 1983,  4:20
FILE NAME (COMMAND)          USER NAME          CPU  % Q WC J/S# STACK   XDS  CODE
-------------------------------------------------------------------------------

*:RUN MIS030.PUB             TEST      .GNET      5  0 C BT S25   4920 31568     0
 :RUN QUERY.PUB.SYS          MGR       .PAYROLL   2  0 C BT S26   5200  3348     0
QUERY    .PUB      .SYS      MGR       .PAYROLL   2  0 C BT S27  16732     0 28172
TEST     .PUB      .PAYROLL  MGR       .PAYROLL  17  0 C BT S6    8212  2380     0
 :SHOWDEV BANKMUX2           ACTUAL    .EPS       3  0 C BT S125  2696 18912  5532
 :RUN SPOOK.PUB.SYS          CONSOLE   .OPERATOR  1  0 C BT S103  4744  6068     0
XXFCS    .EARLE    .EPS      GENERAL   .EPS     252  0 C BT S16  25852  1528     0
XXFCS    .EARLE    .EPS      GROUPX    .EPS     214  0 C BT S73  25852  6220     0
MOO      ?TOOLS    .TECH     BRYAN     .TECH      0  1 C A  S43   8160  2904     1
MBQ      .UTIL     .SYS      MANAGER   .SYS      26  0 D T  J19   5904  3252 12892
XXFCS    .EARLE    .EPS      GROUPM    .EPS     996  0 C BT S119 26620 14960     0
XXFCS    .EARLE    .EPS      GROUPX    .EPS    1016  0 C BT S45  25724 10536     0
RELATE   .PUB      .CRI      ACTUAL    .EPS      50  0 C BT S92  19492  8452     0
QUIZ104  .PUB      .QUASAR   AWAYNE    .LABOR     1  5 C bS S129 12672  1408  5608
TP3000   .PUB      .CCC      ENTRY     .PERSONNL  1  0 C BT S133  9044  6384     0
TP3000   .PUB      .CCC      PSR       .PAYROLL  22  1 C BT S131  8916 15972 12520
MAILROOM.HPMAIL    .SYS      MAILROOM.HPOFFICE    1  0 D P  J41   5212  1864  7452
TEST     .PUB      .PAYROLL  MGR       .PAYROLL 221  1 C BT S11   8340 13732 23976
TEST     .PUB      .PAYROLL  MGR       .PAYROLL  85  1 C BT S9    8340  2380     0
XXFCS    .EARLE    .EPS      SALES     .EPS     401  0 C BT S80  25596 10552     0
 :COBOL PFLSR021.FLSSRCE,U   MGR       .DESIGN    1  0 C BT S140  5296  8492     0
MIS030   .PUB      .GNET     TEST      .GNET     10  0 C A  S25   30796   588     0
-------------------------------------------------------------------------------
TP3000   .PUB      .CCC      MGR       .PAYROLL  DELETED.
```

Figure 6 MOO

```
USER SUMMARY REPORT

                                                                 WORKING SET INFO
PIN USER.ACCT          PROGRAM NAME (command)          CPU  % PRI CSTSZ STKSZ DSTSZ
-------------------------------------------------------------------------------
 32 MGR.PAYROLL        user program file              2237  0 152  7360 16732     0
 33 MGR.PAYROLL        user program file             17415  0 152     0  8212  2132
 59 GENERAL.EPS        user program file            252537  0 152     0 25852     0
 87 MANAGER.SYS        user program file             26308  0 202 12176  5904   768
101 GROUPX.EPS         user program file           1032 S 27 152     0 25724 10536
119 ACTUAL.EPS         user program file             50687  0 152     0 19492  2648
125 AWAYNE.LABOR       user program file              7036  0 153  5608  4656  6664
126 ENTRY.PERSONNL     user program file              1735  0 152     0  9044  6328
131 PSR.PAYROLL        user program file             22876  1 152 12520  8916 15844
138 BRYAN.TECH         OPT.TOOLS.TECH                 637  1 152  1832 11452     0
142 MAILROOM.HPOFFICE  user program file              1231  0 202     0  5212     0
167 MGR.PAYROLL        user program file            222861  2 152 23976  8340 13732
173 MGR.PAYROLL        user program file             86510  2 152     0  8212  2132
175 SALES.EPS          user program file            401055  0 152     0 25596  1176
194 TEST.GNET          user program file             10986  0 200     0 30796   588
```

```
CONTINUE EXECUTION?  (YES/NO)  no
```

Figure 7 OPT

```
 4:24 PM  ELAPSED 00:01:25

                D I S C      W A I T S      Memory       Garbage       Garbage
  Idle   Busy   MAM  User & MAM  User      allocation    collection    allocation
   88%    10%    0%        0%     1%          0%            0%            0

DRIVE-   1    2    3    4
R/sec-   0    0    0    0
W/sec-   0    0    0    0

                                    Q &  %Tot  .....WAIT STATES.....  Disc
 Program name                 J/S#   PIN  Cpu  Abs  Dsc  Bio  Trm  Imp  Pre   IO   Swp  Ovr
QUERY     .PUB       .SYS     S27    C32   0%   0%   0%   0%  99*   0%   0%    0    0    0
XXFCS     .EARLE     .EPS     S16    C59   0%   0%   0%   0%  99*   0%   0%    0    0    0
MBQ       .UTIL      .SYS     J19    D87   0%   0%   0%   0%   0%   0%   0%    0    0    0
SURVEYOR  .TOOLS     .TECH    S43    L90  21%   0%   0%   0%  99*   0%   0%    0    0    0
QEDIT     .PUB       .ROBELLE S25    C91   0%   0%   0%   0%  99*   0%   0%    0    0    0
XXFCS     .EARLE     .EPS     S45    C101 76%   0%   0%   0%   0%   0%   0%    2    0    0
RELATE    .PUB       .CRI     S92    C119  0%   0%   0%   0%  99*   0%   0%    0    0    0
QUIZ104   .PUB       .QUASAR  S129   C125  0%   0%   0%   0%  99*   0%   0%    0    0    0
TP3000    .PUB       .CCC     S133   C126  0%   0%   0%   0%  99*   0%   0%    0    0    0
TP3000    .PUB       .CCC     S131   C131  0%   0%   0%   0%  99*   0%   0%    0    0    0
MAILROOM  .HPMAIL    .SYS     J41    D142  0%   0%   0%   0%   0%   0%   0%    0    0    0
TEST      .PUB       .PAYROLL S9     C173  0%   0%   0%   0%  99*   0%   0%    0    0    0
XXFCS     .EARLE     .EPS     S80    C175  0%   0%   0%   0%  99*   0%   0%    0    0    0
115
```

Figure 8 Surveyor

%Idle	%MMI	%DscI	%Both	%PCPU	%Mam	%Grbg	%Ovhd
93.5	0.0	3.5	0.0	2.7$	0.0	0.0	0.3
97.4	0.0	1.1	0.0	1.3	0.0	0.0	0.2
95.4	0.0	2.1	0.0	2.2	0.0	0.0	0.3
94.9	0.0	1.9	0.0	2.8	0.0	0.0	0.4
61.7	0.0	27.7	0.7	8.1	0.2	0.0	1.8
76.7	0.2	16.8	0.1	4.9	0.1	0.0	1.3
58.8	0.0	22.6	0.2	16.1	0.1	0.0	2.3
15.0	0.0	20.7	2.8	54.0	0.2	0.0	7.5
0.0	1.0	30.2	7.3	53.4	0.5	0.0	8.1
2.2	0.3	31.8	2.2	58.0	0.1	0.0	5.5
1.5	1.5	41.7	2.9	47.6	0.1	0.0	4.8
0.7	0.9	55.7	0.2	38.6	0.0	0.0	3.9
20.7	0.2	51.0	3.9	20.5	0.1	0.0	3.7
40.9	0.0	31.3	0.2	23.9	0.0	0.0	3.7
0.0	0.0	62.2	2.7	28.2	0.1	0.0	6.9
0.2	0.1	65.0	2.6	26.5	0.2	0.0	5.6
2.3	0.0	51.7	1.4	35.4	0.1	0.0	9.2

```
 26.5   0.2  43.6   1.5  23.9   0.2   0.0   4.3
 36.2   0.2  31.1   1.5  26.5   0.1   0.0   4.2 Totals

[PORPOISE]: e
```

Figure 9 Porpoise

The next item on most of the performance programs is the percent of the CPU a process used during the last interval. An interval is defined in each of the performance programs as either the time between characters input from the keyboard or the configurable time in seconds known as the delay time, whichever is shorter. The delay time is used as a terminal read timeout and will allow the program to wait only the specified amount of time before automatically issuing a carriage return to the pending read. SURVEYOR computes this number by holding the cumulative CPU milliseconds used on the last Measurement and subtracts that number from the cumulative number of CPU milliseconds up to the most recent reading. Both of these numbers come from the Measurement Interface. The difference of these numbers is then divided by the total amount of CPU milliseconds expended on all processes during the interval. This difference is then multiplied by 100 to obtain the percent of total time allocated to processes taken by this process. The other performance programs differ in their division. The others divide the difference of cumulative CPU times by the elapsed clock time during the interval. This procedure gives a substantially different number which is the total amount of time during the interval that the process was being serviced by the CPU. Another difference between SURVEYOR and the others is that the other programs obtain their cumulative CPU times from the PCBX area of each processes stack. The sum of the percent of CPU column in SURVEYOR should always be at or very near 100% while the same sum from the other programs would never be 100% and most times not close.

```
   SURVEYOR:
%CPU = ((Cumulative CPU now - Cumulative CPU last interval) /
        Total CPU spent on processes) * 100.0)
   Other performance programs:
%CPU = ((Cumulative CPU now - Cumulative CPU last interval) /
        Total elapsed time since last interval) * 100.0)
```

The que and priority columns should be discussed together. A general understanding of the MPE que structure is required to understand what is presented in the QUE and Priority columns in the performance programs. The MPE scheduling que allows the System Manager to assign a minimum and maximum priority to the three circular ques, C, D and E. The two linear ques have a fixed priority range of 1 to 100 for A and 100 to 150 for B. (see Figure 10) A low priority number indicates a high priority.

The highest priority process requesting CPU will be serviced the next time the \ dispatcher reviews the scheduling que. Before a process may request the CPU, all of the resources required by that process must be available. This means that any data or code segments needed must be in core. Each time a process uses its slice of CPU time, its priority is decremented until it is at the bottom of its que or until a terminal read is issued. When a terminal read is issued, the process jumps to the top of its que.

A process may be pushed out of its que to a higher priority when a higher priority process is impeded on a resource that your process has locked, most likely a SIR. Assume you are running in the CS que say at priority 160 and have the system directory SIR locked. If you are being serviced by the CPU and a system process running at say priority 50, wants to lock the system

directory SIR, the system will see this conflict and push your priority up to priority 50 until you unlock the system directory SIR. A process with privileged mode capability may also call the GETPRIORITY intrinsic and move itself or one of its son processes to a priority out of its que. These are the only two cases that I know of where a process may be running out of its assigned que. This may explain some situations where

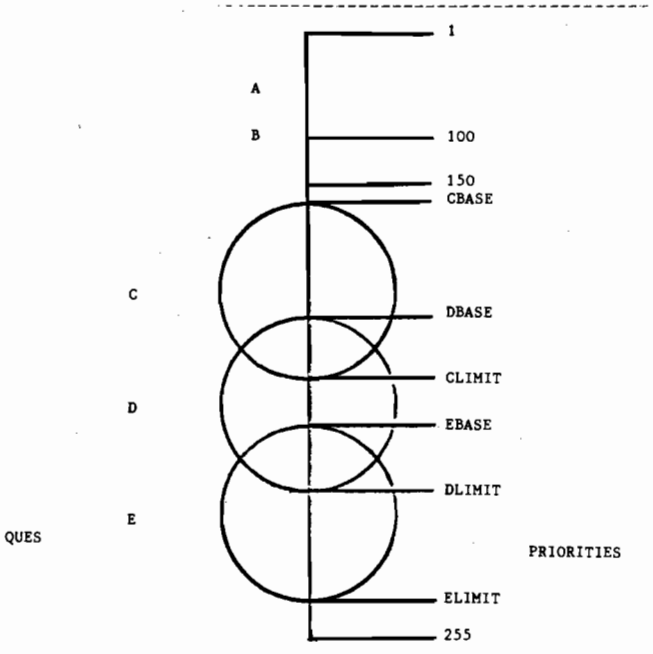


Figure 10

processes in the CS que are running at priority 50.

The que data is kept in the PCB and also in the Measurement Interface. The priority is only kept in the PCB. Surveyor gets its que information from the Measurement Interface while all others obtain their information from the PCB. One version of SOO, version I, could misrepresent itself with its que data. This author apparently realized the potential of a process running out of its assigned que and decided to assign a que based on prioirty. In doing so he fixed the que limits by hard coding them into his program. The limits he decided on are as follows:

Priority	Que
< 150	B (No A que)
151 - 176	C
177 - 201	Sub C
202 - 220	D
221 - 239	Sub D
240 - 249	E
> 250	Sub E

These assignments will most likely be different from the limits set on your machine and consequently the readings from this program will be inaccurate.

The stack size of each process is only found by using the PCB and the DST

tables. First the PCB is searched for the stack DST number (word 3 bits 1:10). The DST table is then searched for this entry. The first word of the DST entry contains the size of the data segment divided by four (4) to save space. The value for stack size, or any segment size, should always be a multiple of four (4).

The current code segment size and the extra data segment size found in SOO and MOO are related. The Segment Locallity List (SLL) is a list of the resources required to be available (in core) for the process to request the CPU. One version of SOO and MOO check the SLL for each process to determine the size of all code segments or extra data segments listed in the SLL. The SLL is a very dynamic table changing every time a process switches code segments, builds a new extra data segment or opens a file requiring a new extra data segment. The data collected from the programs SOO, and MOO from the SLL can be very valuable but it must be understood that these figures will not necessarily include all code segments or extra data segments used by a program, only those required for the process to run.

The program name/command name column found in a version of SOO and MOO requires some knowledge of the command interperter stack. The program name may be obtained from a system routine named PROCFILE. The currently active PIN number is required as input to the routine. The most recently executed command in a command interperter process is a little more difficult to obtain. This command name is stored at DB+1 in the command interpreter stack and may be changed by future releases of the operating system. It is a helpful bit of information for those programs that wish to find it.

Disc I/O information is only provided at the process level by Surveyor and can be very valuable. This information is collected from the Measurement Interface and is persented in terms of disc I/O's per elapsed second.

V. Subjective Comments About Performance Programs

SOO - Version 1

Version One of SOO is a well written program. Block comments are used prior to most routines and do a satisfactory job of explaining the objectives of the routines. The program is very readable and could probably be supported fairly well by someone familiar with SPL. The program is clean in its design and runs fairly efficiently except for the Que column which was described previously. The program tries to derive the Que by obtaining the current absolute priority and trying to fit it to a previously defined Que structure. This could result in substantially inaccurate results in the Que column. The dangerous code is isolated and for the most part is used only when absolutly needed. A good DST retrieval routine is needed to make any of the performance programs work well, and this program has one of the better ones. With the understanding of what the program is trying to accomplish, I feel I could rely on the output of this program with the exception of the Que column.

SOO - Version 2

Version Two of SOO seems to be a predecessor to both Version One and three. This program has many of the same block comments in many of the same named routines but seems to have been written for pre-MPE IV systems. The program only allows for a maximum of 128 process identification numbers (PIN's). The DST retrieval routine is also much longer and more complicated. The routine makes an attempt to go out to virtual memory on its own to retrive data from an absent data segment. I am not sure if this method is still required or not. I have crashed the system several times with a System Failure #16 (DST Violation/Interval Interrupt) trying to access an extra data segment in virtual memory without using disc I/O calls, but there are versions of SOO that seem to work without these calls. I suspect there is an error on my part somewhere, but I havn't found it yet. This program is as reliable as Version One except for

the handling of the Que column. This version takes the que data straight from the PCB entry and does not try to make it fit a predefined Que structure. For pre-MPE IV machines this version would be the best and maybe the only choice.

SOO - Version 3

Version Three of SOO seems to be an enhanced Version One. It contains many of the same routine names and comments but goes a little farther than Version One. This routine is better documented and therefore would be easier and quicker to support. There are some warnings in the compiled version that I obtained but they don't seem to affect the performance of the program. The program replaces the DST retrieval routine used in Version One with a call to another procedure which then calls a system level (undocumented) routine named DMOVE to transfer data from extra data segments to the users stack. I assume that this routine would be more reliable being a system routine but I have no prior experience with the routine. The way in which the programmer implemented the new routine is a little costly in terms of procedure calls but may have been left that way for history. This version seems to be very reliable, and it seems to be remarkably similar to the version of SOO included on the Robelle contributed library.

SOO - Version 4

Version Four of SOO represents a substantially different version of SOO. The program seems to be an entirely new program, and not a clone from some original SOO like the previous three versions seem to be. The program does not jump into a process display but prompts with 'SOO:'. The commands to get the process display are not apparent and it will be initially frustrating for the user expecting the familiar SOO process display. This version provides many more options than are available on other versions of SOO and is aimed at the more involved and sophisticated user. The code is also more complex and harder to read and support. Many higher level constructs, such as using split stack mode to switch to the system globals area, are used making it all but unsupportable by a "mere mortal" programmer. The comments are also rather sparse, but it is a very efficient program. The DST retrieval routine is taken directly from Version One of SOO and seems to be reliable. The program will allow you to do almost anything including running programs like FREE2.PUB.SYS, changing your own priority, and initiating the SEGMENTER? The program has a help (?) facility and each option appears to be safe enough for the average user to try. The program is more difficult to use but allows more flexibility and provides about the same reliability.

MOO

The program MOO claims to be a modified version of SOO and has some similarities. This version is more an extention of SOO than a modified version. The Measurement Interface was incorporated for some I/O statistics and routines were added to allow the running of other programs like Spook. New routines have been added to do things like a FREE2 display or LISTEQ2. The program is fairly well commented, but like Version Four of SOO, the constructs are more difficult to follow. The code is very sharp and efficient, and in my experience it has been very reliable. The program would be difficult to support but it can be done by most mortals. The program only stays inside privileged mode long enough to get what it needs which makes it fairly safe.

Surveyor

Surveyor could have been written as a demonstration program for the Measurement Interface. It follows the constructs of the Measurement Interface very closely. The program is initialized in privileged mode and remains there during its entire execution. This has the potential to cause problems although I have never seen it happen. The program has two displays, either a tables display, similar to Tuner4, or a process display that is very different from SOO, OPT or MOO. The process display is the only display that I have seen which displays both CPU statistics and I/O statistics on the same screen. The program presents the data necessary to determine resource hogs in a fairly concise and readable format. The program uses a large section of the

stack in the DL to DB area to swallow all of the process statistics in one bite while keeping its stack size reasonable (under 10K). The code is a little short on comments but is fairly well structured, and seems to be efficient enough. A modification to the program using only privileged mode when it was required would make me feel more comfortable, but it has not caused us any problems....yet. Since all of the process related data comes directly from the Measurement Interface, I feel I can rely on the data from the program as much as I can rely on the Measurement Interface. I have no reason to doubt the data coming from the Measurement Interface.

Porpoise

Porpoise is another program that could have been written to demonstrate the Global Statistics gathering of the Measurement Interface. The program has a fancy driver but really has only one display and that is global CPU related statistics. The program is excellent for gathering CPU activity to be later consolidated and perhaps graphed. All of the data comes directly from the Global Statistics (Class 0) and is as reliable as the Measurement Interface. The code is clean and short enough to be easily read. Many of the routines used to make the driver fancy are part of the Boeing account library allowing the code to be shorter. The code is very well documented and should be easy to support.

OPT

OPT is Hewlett Packard's On Line Performance Tool written to supply a long needed supported version of a performance monitoring program. OPT is slowly gaining acceptance with the user community as enhancements are made to make it more usable. The Q-MIT version of OPT, version 10, has about equaled what has been available with SOO, MOO and Surveyor all along. Futher comments, analogous to the ones made of the other performance programs, are unavailable at this time because the source has been unavailable.

Summary

All of the programs do a good job at what they were designed to do. I don't think the value of any one of them can be raised above the others without describing specific circumstances. All of the programs with the exception of Porpoise, have no distinguishing marks that could indentify either a company or an individual as the author. Our company has made use of Surveyor, OPT, and a supply of our own home grown utilities to monitor our systems. I trust I have related enough information for the reader to determine which programs are best suited for each situation in his or her shop.

Appendix

*** ATTACHIO ***

```
The  ATTACHIO routine is the primary I/O routine used by
the I/O system.

DOUBLE PROCEDURE ATTACHIO(I'LDEV, I'QMISC, I'DSTX, I'ADDR,
                          I'FUNC, I'COUNT, I'P1, I'P2, I'FLAG);
      VALUE I'LDEV, I'QMISC, I'DSTX, I'ADDR,
            I'FUNC, I'COUNT, I'P1, I'P2, I'FLAG;
      INTEGER I'LDEV, I'QMISC, I'DSTX, I'ADDR,
              I'FUNC, I'COUNT, I'P1, I'P2, I'FLAG;
      OPTION PRIVILEGED, UNCALLABLE, EXTERNAL;
```

*** CHECKLDEV ***

The CHECKLDEV routine does checking on the type of device of a given logical device. (ie. DS Device.)

```
PROCEDURE CHECKLDEV (DEV);
      VALUE DEV;
      INTEGER DEV;
      OPTION EXTERNAL;
```

*** CHECKDISC ***

The CHECKLDEV routine will return global information about the requested disc.

```
PROCEDURE CHECKDISC ( LDEV, INFO );
      VALUE LDEV;
      INTEGER LDEV, INFO;
      OPTION EXTERNAL;
```


*** DELAY ***

The DELAY routine is identical to the PAUSE intrinsic except the parameter is a double word millisecond.

```
PROCEDURE DELAY(TIME);
      VALUE TIME;
      DOUBLE TIME;
      OPTION EXTERNAL;
```

*** DISKSPACE ***

The DISKSPACE routine will return disc free space information about the requested disc.

```
INTEGER PROCEDURE DISKSPACE(LDEV,NSECT,PDISKADR);
      VALUE LDEV,NSECT;
      INTEGER LDEV;
      DOUBLE NSECT,PDISKADR;
      OPTION EXTERNAL;
```

*** DMOVE ***

The DMOVE routine will move data from a users stack to a system extra data segment or from a system extra data segment to the users stack.

```
LOGICAL PROCEDURE DMOVE(DSTTABNUM,TABSTARTPTR,NUMWRDS,USERTABPTR,
      TOFROM,PARMNEQ8);
      VALUE DSTTABNUM,NUMWRDS,PARMNEQ8,TABSTARTPTR,TOFROM,USERTABPTR;
```

```
      LOGICAL DSTTABNUM,TOFROM;
      INTEGER NUMWRDS,PARMNEQ8,TABSTARTPTR,USERTABPTR;
      OPTION EXTERNAL,UNCALLABLE,PRIVILEGED;
```

Appendix

*** FINDDEVICES ***

The FINDDEVICES routine will return all the logical device numbers of a particular type. (ie. Type = 1 will return all logical devices that are discs.)

```
PROCEDURE FINDDEVICES(TYPE,TARGET'ARRAY);
      VALUE TYPE;
      INTEGER TYPE;
      INTEGER ARRAY TARGET'ARRAY;
      OPTION EXTERNAL;
```

*** GENMSG ***

The GENMSG routine will format and display an error message on $STDLIST. Parameter substitution is allow in the message.

```
INTEGER PROCEDURE GENMSG(MSET,MNUM,MASK,P1,P2,P3,P4,P5,
                         DEST,REPLY,OFFSET,DST,CNTRL);
      VALUE   MSET,MNUM,MASK,P1,P2,P3,P4,P5,DEST,REPLY,OFFSET,DST,CNTRL;
      LOGICAL MSET,MNUM,MASK,P1,P2,P3,P4,P5,DEST,REPLY,OFFSET,DST,CNTRL;
      OPTION VARIABLE,EXTERNAL;
```

*** GETSIR ***

The GETSIR routine will lock the requested System Integrity Resource number. The routine will wait for the SIR to become free if it is locked.

```
LOGICAL PROCEDURE GETSIR(SIR);
      VALUE SIR;
      LOGICAL SIR;
      OPTION EXTERNAL;
```

Appendix

*** GETSTATISTICS ***

The GETSTATISTICS routine will return Global statistics information from the Measurement Interface.

```
INTEGER PROCEDURE GETSTATISTICS(CLASS,SUBCLASS,STARTINGITEM,
                                WORDCOUNT,TARGET'ARRAY);
      VALUE CLASS, SUBCLASS,STARTINGITEM,WORDCOUNT;
      INTEGER CLASS, SUBCLASS,STARTINGITEM,WORDCOUNT;
      INTEGER ARRAY TARGET'ARRAY;
      OPTION EXTERNAL;
```

*** GET'PAGE ***

The GET'PAGE routine is a disc paging routine.

```
INTEGER PROCEDURE GET'PAGE(PAGE);
      VALUE PAGE;
      INTEGER PAGE;
      OPTION EXTERNAL;
```

*** GET'DSDEVICE ***

The GET'DSDEVICE routine will return DS information on a given logical device.

```
INTEGER PROCEDURE GET'DSDEVICE (LDEV);
      VALUE LDEV;
      INTEGER LDEV;
      OPTION EXTERNAL;
```

*** GET'DSXREF ***

The GET'DSXREF routine will return a DS device cross reference for a given active process.

```
INTEGER PROCEDURE GET'DSXREF (PIN);
      VALUE PIN;
      INTEGER PIN;
      OPTION EXTERNAL;
```


*** LOCK'DFS'DATA'SEGMENT ***

The LOCK'DFS'DATA'SEGMENT routine will lock the Disc Free Space table.

```
INTEGER PROCEDURE LOCK'DFS'DATA'SEG ( LDEV );
      VALUE LDEV;
      INTEGER LDEV;
      OPTION EXTERNAL;
```

*** PROCFILE ***

The PROCFILE routine will return the fully qualified filename of the executing process.

```
PROCEDURE PROCFILE (PIN, NAME);
      VALUE PIN;
      INTEGER PIN;
      BYTE ARRAY NAME;
      OPTION EXTERNAL;
```

*** RELSIR ***

The RELSIR routine will release a previously locked System Intregrity Number.

```
PROCEDURE RELSIR(SIR,FLAG);
      VALUE SIR,FLAG;
      LOGICAL SIR,FLAG;
      OPTION EXTERNAL;
```

*** RESETCRITICAL ***

The RESETCRITICAL routine will enable the process to abort without causing a system failure should an abnormal situation occur.

```
PROCEDURE RESETCRITICAL(C);
      VALUE C;
      LOGICAL C;
      OPTION EXTERNAL;
```

Appendix

*** RESETDB ***

The RESETDB routine will reset the DB pointer for your process back to your processes stack.

```
PROCEDURE RESETDB (DBX);
      VALUE DBX;
      INTEGER DBX;
      OPTION EXTERNAL;
```

*** SCAN'PAGE ***

The SCAN'PAGE routine will scan a disc page obtained with GET'PAGE.

```
INTEGER PROCEDURE SCAN'PAGE;
      OPTION EXTERNAL;
```

*** SETCRITICAL ***

The SETCRITICAL routine will cause the system to fail if the process meets an unexpected situation which would normally cause the process to abort. This routine is used to ensure all items in a list are completed.

```
LOGICAL PROCEDURE SETCRITICAL;
      OPTION EXTERNAL;
```

*** SETSYSDB ***

The SETSYSDB routine will set the DB pointer for your stack to point to the System Globals area.

```
INTEGER PROCEDURE SETSYSDB;
      OPTION EXTERNAL;
```

Appendix

*** STARTSTATISTICS ***

The STARTSTATISTICS routine will enable any one or all of the statistics gathering types of the Measurement Interface.

```
INTEGER PROCEDURE STARTSTATISTICS(CLASSMASK);
      VALUE CLASSMASK;
      LOGICAL CLASSMASK;
      OPTION EXTERNAL;
```

*** STOPSTATISTICS ***

The STOPSTATISTICS routine will disable any or all of the statistics gathering types of the Measurement Interface.

```
PROCEDURE STOPSTATISTICS(A);
      VALUE A;
      LOGICAL A;
      OPTION EXTERNAL;
```

*** UNLOCK'DSF'DATA'SEG ***

The UNLOCK'DFS'DATA'SEG routine will unlock the Disc Free Space table locked with LOCK'DFS'DATA'SEG.

```
PROCEDURE UNLOCK'DFS'DATA'SEG;
      OPTION EXTERNAL;
```

Bryan Carroll is currently a Hewlett Packard Support Specialist at MA-COM in Burlington, Mass. He graduated from Abilene Christian University with a B.B.A. in Business Computer Science in 1982 and received the Certificated Data Processing (CDP) certificate in October of 1982. Mr. Carroll is also a member of the Association for Computing Machinery and was a member of the Abilene Christian University Programming Team in 1982 winning first place honors at the regional level. His experience includes four years with the HP3000, two of which were at the systems level.

Programming Techniques for Zero-Defect Software

Bruce Toback
Infotek Systems

Introduction

All non-trivial software has bugs.
-- Anon.

Genesis: This oft-quoted saying was quoted once too often to the author by none other than his Hewlett-Packard SE, referring to MPE. Inasmuch as the I am a daily user of this non-trivial software, it occurred to me that perhaps I should examine the underlying foundations of that which is presently the *art* of programming. After all, part of *my* job is to write non-trivial software.

This saying is as much a bias on the part of most programmers and system designers as it is a fact of the computing world today. Is there a cause-and-effect relationship?

It'll be fixed in the next revision.
-- Original author unknown.

Perhaps because software is so "easy" to change, this saying has become almost the watchword of the software industry. Mass-produced hardware products may go through two revisions in a year, and these trivial. Software products may go through two revisions a month, some of them major. This flexibility engenders the "next revision" syndrome: It's OK to leave a problem in the code this time around 'cuz we'll get it next time. And "next time" is only a few weeks away, anyway. Enhancements work this way, of course: enhancements are essentially fixes to problems in the initial system definition. Barring some mutation in the human genotype permitting clairvoyance and/or precognition, the need for enhancements will always exist; there will likely also exist some doubt as to whether a particular change constitutes a "fix" ("it doesn't perform the way I said it should") or an "enhancement" ("it doesn't perform the way I really need it to [now]"). (Anybody who has had their "Problem Report" come back marked "Enhancement Request" will understand this dichotomy.)

Get it right the first time. --
Everybody at one time or another

I have yet to encounter a professional - *any* professional - who does not quote this saying frequently. It tends to become less frequently quoted as deadlines approach, particularly software release deadlines. Strict adherence to this principle need not mean missing deadlines: one need only bend the definition of "right" slightly to insure that this does not happen. "Enhancements" change something over which the programmer has little control: the definition of a defect. (Actually, politically savvy programmers can get quite a bit of control over the definition of a defect, but that is a subject for another paper.) "Fixes" change *only* those things over which the programmer has control. Fixes should always be right the first time.

The phrase "first time," as used in this paper, needs some clarification. An iterative process is inherent in almost any creative endeavor. (Programming, for better or worse, is still a creative endeavor.) "First time," as used here, means the first time anyone except the programmer is asked to use or evaluate the product. ("Programmer," as used here, means not only the individual actually doing the writing, but managerial support, librarians, and others whose primary task is to create *and support* the software products that others will use.)

Zero defect software requires not only good program construction, but good problem definition, testing, and repair facilities as well. Systems should be capable of tolerating program errors in one module without causing a "defect" visible to the user. In addition, with the use of redundant design and checking, systems can and should be capable of repairing problems caused by some program(ming) errors. (The era of the "no-flat" program is at hand.)

A Word about Modules

Definition

Many definitions of a software "module" have been used in the past, with the definition usually set up to suit the point being made. For this paper, a "module" will be a piece of code with a single well-defined function, and a set of well-defined input and output variables. Input and output variables are not limited to function arguments, of course: they include data set and file records, user-input parameters, forms, and other entities that the program can manipulate. In FORTRAN, PASCAL, and SPL, input and output variables include global or COMMON area values that are used or modified.

This paper will not specify the *form* of the module: it could be an entire program, a lexically defined unit such as a procedure or subroutine, or simply a particular group of program lines.

Understandable function

Relative to writing zero-defect software, the *primary* purpose of a module is to provide a well-defined function that can be tested thoroughly. It is almost impossible to test a piece of a program, regardless of its lexical structure, whose function is poorly defined. By doing this, once a piece of code has been determined to perform its specified function, it can be used forever as a "black box:" its purpose *and its effects outside its defined purpose* will be dependable and well-understood.

In-line testing

Testing is important because no technique published to date results in perfect code the first time. (Programmers, like engineers, build and perfect prototypes of their designs before (one hopes) committing them to production. A successful zero-defect methodology will recognize this and not rely on first-time perfection.)

One function of a module is to provide a suitable unit for *in-line testing*. In-line testing of a large, straight-line program is very difficult, since a the function of any lexical entity larger than a statement is hidden in the oversized structure of the program. (Which part of the spider web is *this* thread holding together?) Some programmers, when faced with an oversized structure, determine that in-line testing is useless, and fall back on the technique of repeatedly removing program lines and rerunning the program until they come up with an understandable result, then replacing lines until they understand the function of each removed line and find the faulty one (or ones). Carried to its extreme, this technique leads to rewriting the entire "module." If the same structure is used in rewriting as was used in the original writing, a career loop results.

Interpretation of Test Results

If a piece of a program performs an incomprehensible function, or has too many inputs and outputs, interpretation of test results becomes nearly impossible. This is because one technique of testing may be to vary one of the inputs and watch the results on the output. If 53 outputs must be watched, and their movements interpreted in light of the single input change, one may get the feeling that a certain amount of random variation is involved. In such situations, the tester tends to get the feeling that "this output *looks* right; it must be OK" and the test session ends. If inputs and outputs are ill-defined, this latter test method is often the only usable one.

Complete Testing

A piece of code with an ill-defined function tends to inhibit complete testing of itself, since it is difficult to know exactly what constitutes "complete" testing of something whose function is unknown. In addition, a set of input or output variables that is too large inhibits complete testing, simply because testing all possible combinations of inputs (including those which the definition of "right" precludes as illegal) will be too time-consuming. (The assumption is that some deadline eventually needs to be met.) With a small enough set of input and output variables, and a sufficiently well-defined function, not only can all possible combinations of inputs be tested, but all possible values as well. (In practice, modules for which all possible input values can be tested tend to be trivial, since the testing program needs to be at least as complex as the module under test in order to know all the right answers!)

BITE

BITE is an acronym from the ever-fertile acronym fields of defense hardware. It stands for Built-In Test Equipment, and provides a way of testing equipment without special additional equipment. For hardware, the built-in test equipment generally consists of some generalized testing tools such as multimeters, and some specialized testing tools designed to measure critical operating parameters of the equipment being used.

The same concept can be applied to software: built-in test "equipment" can be included in production programs, to be activated by a programmer, a tester, a support person, or (perish the thought) a customer under the direction of a support person. The test function can be used in three ways. First, of course, it can be used to debug a prototype module before it is released to production. Second, it can be used during system integration to determine the location of a problem, both to the module involved, and to the portion of the module which may be in error. Finally, it can be used to diagnose production programs which have failed in some fashion, either by producing "incorrect" answers, or by unexpectedly aborting. (More on "expected" aborts shortly.) BITE can come in either of two varieties: monitor and diagnostic.

Mutually Suspicion

The concept of mutually-suspicious modules is a general rule in operating system design, where reliability and security are of paramount importance. (This is an example of a self-fulfilling expectation: everybody *expects* operating systems to be absolutely reliable and secure. Why not all application systems as well?) Mutual suspicion means that modules do not rely on their callers for input checking (is the date being sent to the date conversion routine valid?), nor on the modules they call for output checking (does the record you [called routine] said I [calling routine] wanted really exist?). Monitor-type test "equipment" is used to implement this kind of mutual suspicion.

Monitors

In general, pieces of any system should attempt to monitor other parts of the same system. Monitors can be divided into two rough categories, cheap and expensive. The activities being monitored can also be divided into two rough categories: critical and non-critical.

Cheap monitors are those which cost almost nothing to use. (They may take considerable effort to implement, although this is unusual for a **really** cheap monitor.) An example of this kind of monitor is a statement which checks to see if the program is about to divide by zero. Division by zero will (or should) almost always result in an unexpected program abort because it is mathematically undefined. In most cases, the real-world operation being represented by the division is also undefined if its divisor is zero. (What is the cost per item if you don't have any items?) A monitor will check for this and perform any appropriate action. Remember that it is not enough to assume that the system definition (the definition of "right") precludes this: the program in this case should check the system definition. Cheap monitors monitor an operation *every* time it is performed.

Expensive monitors are those which can cost a lot to use. Examples of this kind of monitor are those which check to insure that a summary field in a master record matches the total of the individual detail records. Naturally, the module which takes care of the detail records *always* takes care of the summary. Doesn't it? One way to check this is to call another module which performs the *very* well-defined operation of adding all the details and comparing it to the summary. But since this may involve going through a large number of database operations, it is an expensive monitor: not one which you want an interactive (and impatient) user to wait through every time he or she enters a purchase order item, for example. Expensive monitors therefore need to be "triggered" by some event, either from within the system or outside it. Sometimes the monitor may be outside the software, as when the computer operator instructed to run a data base integrity check every day just before backup. Indeed, "expensive" monitors should always be triggerable manually, just as a check on automatic triggers.

Monitors can be triggered by the system itself in a number of ways. One way is to trigger the monitor every time an infrequent operation is performed. For example, in the MPM/3000 manufacturing system, a check of inventory summaries for a part is triggered every time the user does a physical count on that part. The trigger may be periodic, i.e., every *n* times an operation is performed, the monitor is triggered. The monitor may be probabilistic, i.e., triggered 5% of the time a particular operation is performed. Finally, the monitor may be time-triggered: run every hour, for example.

Very expensive monitors generally need to be controlled by some of the same techniques used to control diagnostic test routines, in addition to their normal triggers.

Damage reporting

What happens if a monitor determines that an error has occurred? This usually depends on the severity of the error and its likely effect on the remainder of the system.

In all cases, the failure should be logged. This can be done using the MPE logging system, or the circular file (as distinguished from the more usual "round file") facility provided by MPE and some other operating systems. The implication is that even if an error is determined to be correctable, its existance must be logged: there *is* a defect in the software, even if it didn't cause a defect in overall system operation.

Obviously, correcting the error is the most desirable option. This implies that one part of the system is deemed by the system designer to be more trustworthy than others (this situation rarely occurs), or is defined by others. Examples of the latter include recording the total amount of a purchase order: the total field is *always* defined by the sum of the amounts in each line item, regardless of whether the amount in each line item is correct. (If this were not so, correcting the line item error would cause a problem with the summary.) Correcting the error should be done *only* if determination of the correct value is not more complex (or identical in method or complexity) than the process which originally generated the error!

Sometimes correcting the error is not possible. This is usually the case with "cheap" monitors: the monitor would not have caught the error if the software which supplied the parameters to the monitor had been working. In this case, the designer, and sometimes the software, must make a determination as to the consequences of continued operation with the data error in place. Sometimes, a default value can be substituted (i.e., a divide-by-zero caught in the act can be made to have a result of zero) that will be as acceptable as the original data. Other times, part or all of the system must shut itself down (gracefully), let the user know what has happened, save as much context as possible for later analysis, and then terminate. This is where a problem detected by the monitor is declared critical or non-critical. A non-critical problem may simply shut down a single function of an overall system, e.g., it may prevent further entry of purchase orders. A critical problem will require shutdown of the entire system. Note that at this point, the zero-defect methodology is no longer involved with preventing defects; it is concerned now with damage control.

Diagnostics

The other type of built-in test "equipment" is the diagnostic. In some sense, the compilers and the operating system already provide diagnostic tools: when a program aborts, MPE prints the location within the program at which the abort occurred, and if asked, will display the data in the program's stack at the time of the abort.

Several problems arise when using the "default" diagnostics. First, they are *very* unfriendly: the "stack display" is recognized throughout the HP3000 world as a sign that the programmer blew it. Second, they are not very informative: they can be interpreted *only* by the programmer, if at all, and present the death of the program as a singular event, without even an invitation to autopsy. Finally, they are useful *only* in the event of a program abort; they cannot be used to diagnose a logic error.

If you are one of the few users that program in FORTRAN or SPL, **HP** provides a subsystem called TRACE/3000, invoked through the use of the $TRACE compiler command. If you know about this subsystem, you are one of a select few: many HP SE's are not even aware of its existence. TRACE/3000 was created in the days when the HP3000 was to be an engineering number-crunching machine, and FORTRAN was to be the primary language. Things have not quite turned out that way. TRACE/3000 is documented in the TRACE/3000 manual. It is not terribly useful, with an obtuse command language and an uneducated support staff. It does not operate with COBOL, COBOLII, RPG, BASIC, APL, PASCAL, or TRANSACT.

Program tracing outside of TRACE/3000, however, is a common and useful technique. Typically, though, there are a number of disadvantages to its use. First, trace messages can typically be activated and deactivated only at compile time. In some languages, the only way to do this is to comment out or physically remove the statements. Commenting is not a well-controlled process, and physical removal of the trace logic is not conducive to retaining the same set of trace messages the next time the module is tested. A better way is to use conditional compilation (for any language except BASIC) to add and remove tracing statements (and other diagnostics as well). **HP**'s compilers have two commands that implement conditional compilation. (Remember that a compiler command tells the compiler to perform some action itself, and does not directly affect the operation of your program. Examples of compiler commands are $TITLE (tells the compiler to title your listing), $PAGE (tells the compiler to skip a page and perhaps add a new title), and $CONTROL (tells the compiler to perform various other actions.)

The compilers provide ten *switches* which can be set and tested by the compiler. These are typically used not only for controlling diagnostics, but for including or not including features in a software product. Compiler switches are named X*n*, where *n* is a digit from zero to nine. They are set with the $SET compiler command:

```
$SET X1 = ON
```

or

```
$SET X2 = OFF, X5 = ON
```

These settings can then be tested (by the compiler, remember) with the $IF command:

```
$SET X1 = ON
  .
  .
$IF X1 = ON
        DISPLAY "ADD-ITEM: TOTAL-COST is now ",
                TOTAL-COST.
$IF
```

In COBOL, this may be redundant, because the PROCESS DEBUG statement may be used. The method works well for other languages, however, as well as for COBOL implementations in which PROCESS DEBUG is not included.

The problem with this type of tracing is that it is available only after recompilation of the system. This is fine *if* there will be an opportunity to duplicate the problem with a different program. During testing, this method of debugging is not particularly desirable, since either two versions of object code (one with and one without the trace messages) must be maintained, or alternatively the program must be recompiled (possibly after changing the compile switches) in order to trace its activity. In addition, in some cases the trace messages themselves may have an effect on the execution of the program.

A method through which diagnostics may be selectively enabled or disabled at run-time seems a better alternative. (You might still want to eliminate the diagnostics in the "production" version to save object code space, but this may be of limited value.) One way to do this is with a debug flag that is set by the INFO or PARM parameters on the RUN command. The program can then test the debug flag to determine whether or not to print trace messages:

```
77  TRACE-FLAG  PIC S9(4) USAGE COMP.
  .
  .
   CALL "GETPARM" USING TRACE-FLAG.
  .
  .
   IF TRACE-FLAG IS NOT ZERO
      DISPLAY "ADD-ITEM: TOTAL-AMOUNT now ",
              TOTAL-AMOUNT.
```

GETPARM is a library routine which obtains the run PARM for a program. (It is available in the contributed library.)

This can be extended to provide multiple levels of tracing:

```
   IF TRACE-FLAG IS NOT ZERO
      DISPLAY "ADD-ITEM: TOTAL-AMOUNT now",
              TOTAL-AMOUNT.
   IF TRACE-FLAG IS GREATER THAN 3
      DISPLAY "ADD-ITEM: Item amount was ",
      ITEM-AMOUNT.
```

Thus, the programmer (or the tester) can select a particular level of trace detail. (The programmer must, of course, be fairly astute in determining what "appropriate levels" should be.) This greatly facilitates the use of tracing during the "prototyping" and system test phases of program development.

(Hewlett-Packard uses this method for V/3000 and some other products: try running FORMSPEC with PARM=1 sometime.)

One problem with this method of doing things is that all modules are traced at a given level, when in fact we are usually (one hopes always) concerned with tracing a specific module. We may in fact wish to use trace messages from more than one module in order to check inputs to and outputs from the module under test. One way to do this is by having a separate trace flag for every module. The problem, of course, is assigning values to all these trace flags when the program is run. Other than simply asking the user/tester about it from within the module under test, is there a way to do this?

By using JCW's (job control words), we can set up *symbolically* a separate trace flag for each module independently. For those modules we don't want to trace, no job control word is set up:

```
:SETJCW ADDITEM=1
:SETJCW DISPLAYORDER=4
:RUN POSYS
```

In the program, then, ADD-ITEM and DISPLAY-ORDER can interrogate job control words ADDITEM and DISPLAYORDER, respectively, to see if they should be traced. This is done with the FINDJCW intrinsic. (The default, if no JCW is set, will usually be "don't trace.")

```
ADD-ITEM.
      MOVE "ADDITEM " TO JCW-NAME.
      CALL "FINDJCW" USING JCW-NAME,
           TRACE-FLAG,
           JCW-STATUS.
      IF JCW-STATUS IS NOT ZERO
           MOVE 0 TO TRACE-FLAG.
```

A similar set of statements at the beginning of each module can be used to define the trace level for that module. (The example, by the way, is contrary to the principles of Zero-Defect Programming. Can you see why?)

By using this method, then, the programmer, the tester, or (perish the thought) the user can selectively activate various levels of tracing - of diagnostics - to determine what the program is doing at any given time.

Symbolic Debugging

Symbolic debugging techniques such as tracing can be extended to include other concepts. Using a control-Y trap, it is possible to allow the programmer or tester to execute a special purpose debugging subsystem. Such a subsystem may be command-driven and may be used to print various record values, to display the status of open files, or to run various "expensive" monitors on demand. For large systems, this technique is invaluable and well worth the amount of work it takes to implement. Typically, such a subsystem will require between five and 10 percent of the code required for the entire system.

Conclusion

Program bugs need not be a part of every non-trivial software system. With proper care in construction of the software, and with the assistance of some simple programming and testing assists, bug-free software can be delivered to the end user every time. (I recognize that some of these methods are impractical for use in debugging interrupt-driven I/O drivers, but very few applications include these!)

To summarize the techniques presented here:

1. Divide your programs into well-defined modules - logically if not lexically. Make sure that each has a name.
2. Include "cheap" monitors to check inputs to each module every time it is used.
3. Include "expensive" monitors to check system data base integrity at intervals as frequent as the user will tolerate.
4. Provide for a method for logging errors discovered by your monitors. Use the softest failure mode you can without jeopardizing system integrity.
5. Provide a method for repairing as well as logging all repairable errors.
6. Provide a trace facility that can be used by the programmer, the tester, the support person, or (perish the thought) the user. Try to make the facility usable "on demand," rather than waiting for specific action by a programmer or librarian.
7. If your system is large enough, provide a debugging subsystem to print values of various records, to run your "expensive" monitors on demand, and to print information about execution of the program. Make this subsystem accessable on demand if possible.
8. Don't be satisfied with fixing it in the next release!

Bringing Business Graphics to the Masses

by Alan Arens
Sales Manager

Introduction

What comes to mind when one hears the expression "computer graphics"? Today, many people feel that computer graphics falls into one of several groups:

1) Simulation Graphics

2) Scientific/Engineering Graphics

3) Analytical Graphics

4) Computer Aided Design/Computer Aided Manufacturing

5) Command and Control Graphics

6) Business Graphics

All of these different forms of graphics have as their main goal the need to quickly transfer information to the user or reader of the graphics. The first five groups have one major difference with business graphics. The readers of these five groups are specialists, meaning that in a sense they are trained to read the graphic images presented to them. This however, is not true of business graphics. Each graph, picture, or chart in business graphics must be totally self descriptive. The engineer's blueprint is the antithesis of business graphics.

The principle use of business graphics is to transfer information. Thus the extensive use of graphics at meetings and presentations fall into the category of business graphics. This paper is an overview of what is needed to bring business graphics out of the artists' shop and into the business office by using computers and their associated peripheral equipment.

A Brief History

The first major introduction of computer based business graphics was in 1970, when Integrated Software Systems Corporation (ISSCO) introduced their package of computer programs called DISSPLA. It was a major improvement over what was then available. Limited libraries of subroutines were available to insulate FORTRAN users from the character string commands to output devices.

DISSPLA was a mojor breakthrough in computer graphics but it still had major drawbacks. Firstly, it was designed only for large mainframe systems which could handle large programs. Secondly, the DISSPLA price was not competitive with the artists' shop. Thirdly, the graphics were not of presentation quality at that time. Lastly a FORTRAN programmer was needed to write a computer program for each chart since DISSPLA was a collection of subroutines with no main line.

Since the late 1970's, many other graphics packages have been introduced. Many of these packages have overcome the short commings of that early DISSPLA package and run on much smaller machines. Nearly all have a user interface that does not require a programmer's background. In this same time frame, devices of desktop size have been introduced to display or draw the charts. Tektronix was the first major manufacturer to sell graphics terminals. To this day Tektronix has the largest number of graphic terminals in the field with respect to quantity and different types. Hewlett-Packard was the first to introduce a one-mil (0.001 inch) resolution multicolor desktop flat bed plotter for less than five thousand dollars. They are still considered by many to be the leader in desktop plotter manufacturers.

The User

The top level management has known and used business graphics for a long time. However, since each chart required the skill of the graphics artist, and this was expensive, the lower levels of management could not justify the expenditures of time and money to have graphic artists make their charts. Thus the Orator Ball on the IBM Selectric typewriter was middle management's tool for one form of business graphics and/or word charts. The

major task for middle management is to get ideas across, not to use fancy charts. Computer generated business graphics appears to be part of the answer of getting these ideas across. Over the last several years costs of computer generated business graphics has steadily decreased. There is now software and hardware in reach of almost every budget. The quality of the graphics produced using computers has improved to the point of being used by upper management and in the board room.

When top level management was using charts made by graphic artists, there was a distinction between the user (top management) and the creator (graphic artist). Today with computer generated business graphics, the user and the creator, is in many cases the same person. Thus, the range of users of graphic packages is vast. On one hand, you will have the modern day graphics artist who makes his living using the graphics package every day. On the other hand, you have the user, who once every few months, wants to make a set of charts. The business graphics package must fulfill the requirements of both users. The capabilities of the package might be forgotten by the occassional user and he must be reminded of them. The capabilities of the package must be extensive. Otherwise, the manager might request a chart to be created which is beyond the capabilities of the graphics package.

There are roughly three levels of sophistication for users of computer generated business graphics. At the basic level, the user has his data on a sheet of paper and wants to see it in some graphic form. He may know that it should be in the form of a pie, bar, or line chart. Other than that, he has no partiular desires as far as the graphic layout is concerned. He might know what size paper he wants. At the other end of the spectrum, the user knows exactly what he wants the chart to look like. Almost no variance from the picture in his mind's eye will be allowed.

In the middle, the user has some specific ideas on what he wants the chart to look like, but at the same time he is willing to compromise, in order to get his chart as quickly as possible. He might want the color of a bar to be solid red, but would not settle for a hatched red fill. Business computer graphics are designed for this middle group where a vast majority of users are to be found.

If the user is the modern day graphics artist, he is the important link between the computer business graphic system and the finished charts that someone else needs. For the graphics artist who uses the system daily, wants the chart design session to be brief. He does not need detailed on-line instructions. On the other hand, if the user needs to create fifteen charts for his tomorrow morning meeting and it is now afternoon, the graphic system must allow this occassional user to meet his needs in a timely manner. If the user has not used the system regularly, he proably will not have time to refamiliarize himself with the package. If the chart cannot be done quickly, then the user will become frustrated. Naturally if the chart cannot be done quickly, it will no doubt be to late to do the fifteen charts. Thus, one must require that the system be easy enough to use to get an acceptable chart finished in an acceptable amount of time.

Output Devices

There are five basic types of output devices; plotters, graphic terminals, film cameras, laser printers, and ink-jet printers. Today the plotter is proably the most common hard copy device, but the multicolor ink-jet printer is gaining in market share.

Plotters are devices that draw with pens on paper, film or other transparent, translucent or opaque media. Plotters come in three forms, flatbed, drum, and a cross between the flatbed and the drum where sheet stock is held between special edge rollers. Flatbed plotters may have the capability of using roll fed paper, not unlike the drum plotter. Flatbed plotters have the ability to handle thick stock such as bristleboard. Drum plotters have the advantage of being able to draw a chart up to 120 feet long. The plotters that are a cross between these two give one the advantage of a D or E sized plotter capability for far less cost than a D or E sized flatbed plotter. These plotters are now available in A and B sizes for less then 2,000 dollars.

Graphics CRT terminals are available in both black and white and color. The recent decrease in computer memory costs has significantly decreased the cost of these devices. Like newspaper photos, a graphics terminal represents the picture by a field of dots or pixels. The more pixels per inch, the better the picture. The quality of the output still does not begin to match a plotter, since a plotter has the equivalent of 1,000 pixels per inch whereas the best available graphic CRT is 1,280 pixels across the whole 19 inch screen. One can expect to see 4,000 pixels on the x-axis in a few years, after the new 256k memory chips become low cost items. With present day CRT's a line that is not perfectly horizontal or vertical will have a step or jag somewhere along the line as one moves from one row or column of pixels to the adjacent row or column. With a limited number of dots, small incremental line segments may fall between pixels and never show. To make small text readable, the number of pixels must be quite large in both directions.

All modern day graphics CRT's store the contents of the pixels in a special memory. This memory is read sequentialy to generate a video

signal that is painted on the screen. There are many devices that can use this video including cameras, video projectors and in the last year, low cost ink-jet printers. Nearly all of these devices have a resolution much greater than the pixel memory of the terminal. Thus the graphics terminal is the limiting factor. Matrix recently announced a product that by passes this resolution problem and generates a 4,000 x 4,000 pixel color photograph or slide.

The last device to be considered is commonly called a laser printer. Here the computer stores the pixels in its memory. Hewlett-Packard's laser printer is an example of this type of printer.

Graphics Software

There are two steps needed to create business graphics. The first step is to "design" the chart by generating the specifications and the data. The second step is to take this data in the form of a file and "drawing" the chart on the output device. Programs that allow users to interactivly draw on the screen, while certainly are graphics, are not included. The most common hard copy device is still a plotter, not a pixel memory dump device which would be needed to catch the information that is only on the screen.

The form of the interface between the user and the design phase of the program is important. There are four forms of this man/machine interface to be considered:

1) **Programmatic**

A programmatic interface is a set of subroutines that are given to a programmer so that he may write an application program to create the charts and graphs. The advantage of this method is that if one has a very specific application and there is no off-the-shelf software, then a program can be written to fulfill the requirement. The disadvantage is that if one has many different types of general purpose charts, the programmer will need to create all of the programs required to create the charts.

2) **Command Driven**

A command driven interface is where the user enters all of the information directly into a file with no prompting. The major advantage of a command driven interface is that if one is knowledgable, one can very quickly define his chart without waiting for questions or screens. The major disadvantage is that a novice user can get completely lost.

3) **Screen Menus**

A screen menu is nothing more than a V/PLUS screen or an equivalent type of formatted screen. The advantage of screen menus is that a chart can be made by just filling in the blanks while jumping from screen to screen. However, if too much screen switching is needed then one may get lost. Working with screens can slow one down since one must switch to another screen to get additional information if one does not understand the selection presented. A major disadvantage of screen menus is the limited number terminals that are usable.

4) **Question and Answer**

A question and answer interface is just that. The sytem asks a question and then waits for the user to answer. The major advantage of this interface is that enough information can be presented so that if the user is a novice the question can still be answered correctly. The major disadvantage of question and answer sessions are when the user is an experienced user. He only occasionally has a need for all the information presented to him. Thus a questions and answer interface, to be effective, must allow the user to select, in a user changeable manner, the amount of information he needs. With short questions the experienced user can design the graphics faster.

Experience has shown that if the total number of choices is below some threshold value, and the user uses the system frequently, then screen menus are best. If the number of choices are above this threshold, and the user may only use the system, for example, for quarterly reports then the question and answer interface is better.

Requirements for a Graphics Package

Firstly, the package must be capable of drawing a large percentage of the charts required. No package will do everything. The package must have the capability to handle bar, line, pie and word charts in a rapid manner either via menu screens or a question and answer session.

Secondly, it must be cost effective. The graphics package must create charts more economically than an artist can do it by hand. Even if special training is needed for the modern graphic artist, costs will normally be much lower since much less of the artist's time is needed on a properly designed graphics system. If all one needs is to see if the line goes up or down, the user's Apple or IBM PC will proably suffice. However, if one does need better lettering, shading, and other more complex features, then a more powerful package is needed.

Thirdly, the graphics system should have easy access to the data. If one has only ten data points, the user should be able to enter data from the terminal keyboard. If on the other hand one has 100 or 1,000 points then the graphics package should be able to read the data from a file either during the design step or the draw step.

Lastly, if the charts are for formal presentation, or an outside client, other capabilities must be considered:

1) The capability of using high quality devices for high quality output.

2) The ability of switching from a horizontal format on an overhead transparency for a presentation, to a vertical format on paper for a report.

3) The capability to draw to film on a plotter without allowing the colors to run together or "bleed".

4) The ability to have enough annotation (title lines, legends, notes, etc.) so the chart does not need to be modified by hand or by another program.

5) The capability of having enough different shading patterns so the chart can stand alone in black and white as well as in color.

6) The use of proportionally spaced characters with the additional ability of displaying data in straight columns.

Summary

There are a few quesitons which one must answer before one purchases hardware and software to produce business graphics with a computer. First, you must evaluate all graphics that you are planning to automate. Are most of the charts very different or quite complex? Is it possible that an off-the-shelf computer graphics package might not meet your requirements? It is even possible that computer graphics might not do the job in a cost effective manner with the development effort involved.

If your charts are similiar to each other, or if the charts do not have much variation from simple line charts, bar charts, or pie charts, then computer graphics may simplify ones work. The next step is to find out what level of sophistication, the software needs to produce the charts. Are the charts needed for top managemnt, and therefore, must be of top quality? Are the charts used for briefing colleagues, and can have computer looking "stick" characters and single stroke lines? If the later is all one needs then graphics software on most microcomputers will suffice. If the answer is yes to both questions, then one needs a package with much more power and flexibility. If one has a limited set of specific charts, a software package which allows creation of a specifications file with later quick data addition would be helpful. A specifications file defines the layout of the chart (number of bars, wedge shading, etc.) Then all one needs to do is enter the data and one has his finished graph. This is also the situation where a subroutine library is most useful. If a programmer is available, programs can be written to create exactly the few charts needed with flexibility only where it is wanted without the overhead of more general graphics packages.

Once one has decided what capabilities the software needs to produce the required charts, the hardware will fall into place. If one needs to produce 35mm slides, one will need a graphics terminal with a camera. If overhead transparencies are needed, then a pen plotter will do nicely.

As one decides what software and hardware one needs, one must remember what personnel will be operating the software. Since they will be the key to success using computer graphics, one should allow them to work with the software before a purchase is made. Many companies will give a trial period at a reasonable cost, to allow for such an evaluation. If the users do not like the software or hardware, the probability of getting a second chance with another system is almost zero.

Business graphics are part of the rapidly growing computer graphics industry. New hardware, software and systems are introduced continuously by many vendors at all levels of performance. "A picture is worth a thousand words" is as true in today's business enviroment as anywhere else.

Alan Alexander Arens was born on June 5, 1959 in Wichita, Kansas.

While attending college, Alan worked for Hewlett-Packard in the Eastern Regional Sales Center in the training center. While at HP, Alan became familiar with the HP 1000 & HP 3000 computer systems.

Subsequently, Alan went to work for Arens Applied Electromagnetics Inc., as a staff programmer. While working on custom software he assisted in the development of the early versions of what is now the company's major off-the-shelf product --- PRESENTATION GRAPHICS by ARENS.

As the company expanded, Alan moved from his programmer's position to become programming manager. This past June, Alan Arens was moved up to the position of Marketing Manager.

Configuring DSN/MRJE for Simplicity and Savings

by Karl C. Collins

Introduction

An indepth analysis of our MRJE operations re opportunities to simplify the operator intervent while reducing costs associated with transmissio waste, and technical support. This overview co configurations required by DSN/MRJE (Distrib Network/Multi-leaving Remote Job Entry), an capabilities and consequences of certain config as they pertain to managing output, line efficie simplifying the console operator interface. Spe consideration will be given to handling special non-standard print output. Also, configuration affecting multiple hosts will be discussed.

Historical Background

A little historical background will help illustra problem areas that precipitated our exploring a utilization of the DSN/MRJE capabilities. We HP3000 Series 40 from a card-based batch pro both MRJE capability and a minimal amount o As a single user system, all physical devices wer MRJE whenever the host was online. MRJE ou IBM host until specificly released by the operat spooling of either jobs for transmission or inco multiple copies of reports were produced by tra entire report the appropriate number of times. required additional effort by the operator to m install a carriage control tape, release the file, a forms, etc. This culminated in a great deal of e part of the operator to manage MRJE and the addition, the transmission lines were being used inefficiently due to waiting for operator comm retransmissions , and transmitting at the speed print rather than the speed the modems could conversion to the HP3000 complicated the situ a multiuser environment, spooled printer, and t interface program; this made the operator's job difficult as the printer was no longer dedicated effect, we moved from an inefficient situation inefficient situation.

All of this prompted an effort to find out how to reduce or eliminate the operator interventio the files back from the host and handle them i fashion. Months of trial and effort and searchi knowledgeable users passed before we reached configuration. We realize that there are still ot capabilities of the DSN/MRJE product to be ex explained.

Overview of MRJE

Before going into detail, a brief overview of M MRJE is an advanced form of the RJE (Remote developed by IBM for the submission of batch j from a remote terminal. RJE transfers one file maximum of one printer, punch and card reade terminal. A bi-synch protocol is used where ea transmitted is checked for integrity and ackno next block is transmitted. With MRJE, multipl transferred between multiple host and remote time. MRJE can place both acknowledgements block being transferred and will send different time in different blocks of data. Up to seven p and readers may be configured for a MRJE re

The physical components of DSN/MRJE includ Network Processor) for a HPIB machine or a SS Single Line Controller) for a Series II/III machi to 52Kb synchronous modem, a dial up or dedic corresponding equipment on the host system.

The software components include the user inter (MRJE.PUB.SYS), the line monitor program (M the device drivers for the printer/punches, read console. On the host side the software may be J or ASP.

The system we have is a HP3000 Series 40 wit modem (4800 baud), and a dial up line. Our ho software. The specific details of this paper are towards this configuration and will be differen for other configurations, particularly as regards other than JES2.

Configuration Overview

There are six different components to be config up or changing MRJE. Briefly they are as follo

- o The modem is configured primarily to mat dial up vs. dedicated line. This configurati changed.
- o The INP/SSLC is configured to match the DSN/product. There may be multiple conf INP to match the different products and ty modems.
- o The DSN/MRJE device drivers and spooler punches and readers are configured in the Configuration; there may be multiple DSN configured.
- o The host configuration file (MRJECON) co about each host system and the type and co remote terminal to which the host believes
- o The host system, such as JES2, has a remot configuration describing the type of termin System/360 etc.), the number and type of p their default setup values.
- o The job stream (MRJESTR) that is used to monitoring program also has some configur capabilities. It contains file equations that determine the disposition of the various ou the message log from the host.

INP Configuration

The INP configuration is downloaded to the IN device is opened for use. As a result there may different configurations for one physical INP. configured for different software products suc DSN/MRJE, which use the same modem and li configurations may be for different modems an dial up and dedicated, which will require switc the modems. The attributes of an INP configur changed except via Sysdump or a Coolstart. As difficult to occasionally use MRJE on a dial up normally configured on a dedicated line. The I can be seen in the System I/O configuration an of the SYSINFO utility. Note that in Example are configured (ldevs 15-17) and that they all physical INP, DRT# 82. Naturally with only o several DSN products at the same time.

System I/O Configuration

The system I/O configuration determines the that are used and therefore the pseudo peripher are available for use by MRJE. Of course the h have the same peripheral devices listed in its R configuration. It is possible to have seperate co for different hosts. Several of these pseudo dev required, ldevs 50-53, and 61 in Example 1, tw console, printer, and reader. Any additional rea and punches must also be configured here. It s that the MRJE devices are pseudo or imaginary they may be spooled. Their DRT# refers to the physical printer or other device. It will be possi them to a real device through a file statement i Configuration File. The DRT# also implies a o relationship between the INP configurations an devices; and, as a result, providing both a dial-u line capability to a host will require two sets of

When two hosts use the same configuration, the use the same spooled reader for jobs that are su though a specific host has to be identified befor submitted. When a host comes online the entir spooler will be transmitted to the online host, r where the jobs were supposed to be sent. One s problem is to configure a seperate spooled read and then have the operator suspend the other r bringing a host online. A better solution is to c completely seperate MRJE configuration; this brought online only one reader will be known t accessed. This solution will be transparent to t users. In Example 1 two hosts have been config different number of pseudo printers. Both host indentical INP configurations.

Remote Terminal Configuration

The Host system has a Remote Terminal Config keeps track of what type of remote system it is with. This configuration must be consistent wi devices in the System I/O configuration and th Host Configuration File. Typically DSN/MRJ System/360 with a console, multileaving interf transparency option. The details can be found the DSN/MRJE Reference Manual.

In terms of handling the output the important configuration is the number of additional print devices as well as their default parameters. Ex devices configured for our remote terminal. T punches should be automatically started, have a "STD.", and be in automatic forms mode. In ad specify the outclasses each device will accept a limit. The automatic start option simply means will always be available for MRJE output, rath operator send a command to start the printer. output for a remote terminal, it looks for an av the correct type, outclass, and forms. The type printer/punch, the outclass is one of 36 outclas (A-Z,0-9), and the forms being the correct for device has default value(s) for each of these att are initially assigned when the host system is IP (coldloaded). The remote terminal operator ma to the devices at any time. Optionally, a print can be set to reserve a printer for reports of a s range.

When there is output for an automatic forms valid outclass and a new or different forms na will send a SETUP message for that printer wit name. The host will also reset the printer to th name. DSN/MRJE will acknowledge the new f printer command to the host ($SPRn). When D SETUP message for a non-standard form, it wil matching environment file and, failing that, at message to the file. It will continue to do so fo recieved by that printer until the host is signed SETUP message is recieved. If item 14 of the file is configured incorrectly, DSN/MRJE will SETUP message, and not respond to the host sys terminal operator will then have to send a start ($SPRn) to the Host in order to recieve the out

There is a serious conflict within MRJE concer printers and the SETUP command. The host wi message and change the forms name parameter changes. The printer is reset to the configured only when the host is IPL'd and not when the r signs off. On the other side, DSN/MRJE keeps current form on each printer between changes, when the host is brought online all the remote forms = STD. As a result, the host and DSN/M of sync in terms of what form they believe is as printer. There is no mechanism to bring the pr sync other than manual inspection and interve the operator. Even if DSN/MRJE did keep tra between sign-ons, it would not know when the As a result, using special forms names is not co at this time. iIf this causes serious problems yo the systems programmers at your host site to se patch for JES2, or the appropriate software. T investigating this problem.

A few words are in order concerning the outcla host site assigns their own values to the outclass partial list of outclass codes used by one of our in Example 2. By assigning different sets of ou your output devices, and setting up file equatio device, you can provide for automatic handling of output.

Host Configuration File

DSN/MRJE maintains a file for each host with information about the MRJE pseudo devices, th terminal configuration. This file ties together configurations. As shown in Example 3, the fir host name. DSN/MRJE uses only the first lette thereby allowing up to 26 hosts. All MRJE file specific host end it that letter; for example, MR MRJE host configuration file for the host DAL

Items 2,3 and 19 refer to the MRJE pseudo dev I/O configuration and, indirectly to the INP co

DSN/MRJE scans different parts of the output in order to determine the output type, host job SETUP messages, etc. Items 8-17, and 37 tell look for this information. Configuring these it confusing and, if they are incorrect, the results unusual. Appendix A of the MRJE Reference values for these items; however, various hosts m established values that are at variance to those in the manual. These values should be obtaina technical support staff for that host. The HP p installed DSN/MRJE for us were initially unab correct settings. Perseverence and the willingn trying finally led us to correct values. Should y similar difficulties, examine the print banner a message log carefully and keep trying.

Items 20-35 determine the disposition of print, output. These items can be given values of a de ldev#, or back reference a file equation by use Example 3 shows the use of both a device class equations such as !PART3. Output that is route class or ldev will have the host job name used a Output that is routed with a file equation will specified in the file equation.

When output comes back from a host, DSN/M default, solicited, or unsolicited, in addition to punch or forms. In general default (and uniden goes to the devices specified in items 20 and 28 output goes to the device specified by the user job. Unsolicited output is routed according to t items 21-27 and 29-35 that corresponds to the was select to transmit the data from the host. explanation of these different kinds of output shortly.

The remaining items in the host configuration clear and will have little effect on your ability output from the host.

DSN/MRJE Job Stream

Each host has a job stream that is run in order MRJE line monitor program. The job stream is for host Dallas) and the monitor program is M stream must be in PUB.SYS and run as MANA should be carefully secured to keep the passwor discovered; this also makes sysdump tapes more ever. Example 4 is a listing of MRJESTRD.

There are three entries in the example that des First, a file equation must be present for all soli unsolicited entries that back reference a file eq back references are in the individual Submit co the users and in the Host Configuration File. S directs a copy of all remote terminal console m MRJEMSGD (D for Dallas) that defaults to $N can be used here to redirect that output to the message file, circular file, or for whatever othe may find useful. A listing of the console messa helpful in troubleshooting after some output h a fashion other than you intended. This is espe since you will not see the console messages unle running the MRJE user interface program and terminal console mode at the time the message

or the DSN/MRJE monitor program. And thir in Example 4 are executed after the host is sign check to see if any unexpected punch output a site we deal primarily in print output, punch o about four or five times a year, usually without This keeps it from being printed out on the pri lost, or from just sitting in a disk file unnoticed

Host JCL

When print output is produced on the host it is attributes that determine how the output will first attribute is a single character outclass cod site determines what type of output each outcl represents. The second attribute is a forms na characters long and most often has the default These two parameters or attributes are given in card such as:

```
/*SYSUT1 DD SYSOUT=(F,,F123)
```

where the outclass=F
and the Form name = "F123".

There are also two parameters in the host JCL Job statement that are useful to know. The MSGCLASS parameter is used to assign the outclass of the JCL listing that is produced for the job. This listing is similar to the $STDLIST created for a batch job on the HP3000. Generally this listing is used only when the job has failed for some unknown reason. Furthermore it is rather unintelligable to a non-IBM person. Our host maintains an archive of all JCL listings and so we defer these listings in the spooler and then delete them every evening. This procedure change has saved us in excess of three cases of paper a year for one program alone. The MSGLEVEL parameter in the Job statement is used to indicate the detail level of the JCL listing. To minimize line transmission costs, we have this set at the minimum level. These two parameters have a format as follows:

```
MSGCLASS=P,MSGLEVEL=(0,0)(0,0)
```

This will cause the JCL listing to have an outclass of "P" and only the Job card will be printed (first 0) with no messages unless the job terminates abnormally (second 0).

DSN/MRJE Submit Command

When a user wishes to submit a job for batch processing on the host system the DSN/MRJE user interface program is used to select the host and submit the job. There are several parameters on the Submit command that are crucial in determining how the output of that job will be handled when it is recieved from the host. The format of the Submit command is as follows:

```
SUBMIT filename;PRINT=    ;PUNCH=    ;FORMS=
```

When the Submit command is entered, the input file is translated from ASCII to EBCDIC (unless otherwise instructed) and placed in the spooled reader specified in the Host Configuration File. When the host is next online, the job will be transmitted. Also, at this time an local DSN/MRJE job number is assigned and an entry is made in the host job log (MRJEJOB), along with the parameters. This job log is updated with the host job number and time stamp when the job is actually transmitted. When the output is recieved from the host, this job log entry will be used to determine what parameters were specified for the disposition of the output. The three parameters are optional and will accept two forms of values for a total of three classes or kinds of results. These are referred to a Solicited, Unsolicited, and Default output.

Solicited Output

When a Submit parameter is given the value of an ldev#, device class, or a back referenced file equation, the output is called solicited. When the output is solicited, all the output of that job and type is routed according to the value specified. In this fashion all the print output for a given job can be routed to a given printer with a device class entry, the forms output can be handled by back referencing a file equation, and the punch output can be routed to a disk file via a file equation. The file equations must be in the MRJE execution job stream (MRJESTRD, Example 4); this will require some advance planning as few users will have access to that job stream. The real problem that arises with solicited output is if your job produces several output files of the same type, ie. print output, you will not be able to to handle them differently. This is where unsolicited output has an advantage.

Unsolicited Output

When the parameter given is a zero (PRINT=0), the output is called unsolicited. As such, all the output for that job and type is routed according to the entries in the Host Configuration File associated with the output devices that the Host used to transmit the output to the remote terminal. Specifying unsolicited output may cause different kinds or outclasses of output to be handled differently, even when they come from the same job. Generally, the unsolicited printers are used to handle certain kinds or classes of output, such deferred output, multipart paper output, output for specific printers, etc. The limitation of unsolicited output is that you only have 7 printers and 7 punches with which to work. You may also be constrainted by the outclass codes that are used by your host site. The additional unsolicited printers that are configured into the system are going to cost some additional overhead whenever the host is online and probably should be used only if there is sufficient traffic, especially if you have a dedicated line and keep the host online all the time.

Default Output

When no parameter is given, or if the output cannot be linked to a given job submission, the output is referred to as default output. Default output is routed to the default devices which are specified in items 20 and 28 of the Host Configuration File. There are two cases where default output is a concern. One is where default output is appearing that should have been handled in another fashion - for some reason DSN/MRJE is unable to identify the output and match it with the job that was submitted. To identify the problem, check the Job Log to see if the entry has been purged or lost for some reason. Next, look at the console message file or listing to see how MRJE interpreted the output when it was recieved. The other case concerns output from the host that is being generated by some process other than jobs submitted through MRJE. There is no easy solution in this case as DSN/MRJE has no way of knowing how the output should be handled, and therefore the Default option is used. It is possible to write a program to examine these files and redistribute them according to whatever parameters are appropriate. In a paper given at the Montreal conference, Rolf Frydenburg briefly described a batch program running SPOOK as a son process that periodicly examines the spooler for files to be redistributed.

Environment Files

There is one last way to provide for special handling of files through DSN/MRJE, and that is with environment files. MPE supports environment files for the HP2680 laser printer and the HP2608S line printer. The environment file contains a forms message, cctl information, and, for the laser printer, the image of the form to be printed. DSN/MRJE supports environment files only for the HP2680 laser printer at this time. If you (1) have an environment file, (2) it is placed in the appropriate group and account (@.ENV.HP2680, examples to the contrary in the reference manual are incorrect), (3) it has a four character name that matches the forms name of some host output, (4) the output is returned via an Automatic mode remote printer that generated a SETUP message, and (5) your laser printer has been configured with a device class name of "LPS", then DSN/MRJE will take the output and corresponding environment file and route them to the "LPS". The reference manual says that the SETUP message will be acknowledged by DSN/MRJE only for output that is of outclass = A and sent on remote printer 1. This may or may not be true for environment files, but it is definitely not true for SETUP messages for non-environment forms file output where it works fine for all outclasses and remote printers.

Forms Files

Forms files are sufficiently complex that a summary of their operation may be of help. A forms file is defined to be any host output with a a forms name that is not "STD.". The key to having DSN/MRJE recognize and handle forms files differently from print and punch output is the receipt of the SETUP message from the host. As indicated earlier this message is not completely reliable without manual intervention of some kind or a custom modification to the JES2 software at the host site. Example 2 shows a technique for minimizing the loss of reliability. An outclass code of "F" is used to designate forms output and a special printer (PR6) is configured with a file equation that labels and defers all output. Then if the print output is designated as unsolicited and a SETUP command is not received for a forms output, the file will be handled according to the forms file equation. All our jobs having forms output are submitted with the unsolicited print option. In this fashion, if the SETUP message is lost, the output will be deferred and and labeled to catch the

operator's attention. This technique works well for all forms output created by jobs that are submitted from the HP3000. Forms output from elsewhere still runs the risk of becoming default print output if a SETUP message is not received.

Summary of Output Handling with DSN/MRJE n

Without operator intervention, DSN/MRJE provides the capability to bring back output from a host in two general ways. One is the ability to establish up to seven classes or kinds of output (14 with print and punch output), each of which will be handled in a specified way. This is done through the use of the unsolicited designations and outclasses for the different devices and types of files. The other way is a variety of parameters that allow individual jobs or files to be indentified and handled in a specific way. This is done through the use of a solicited designation or a nonstandard forms name. The forms name can be used to single out a file with an environment file or the FORMS parameter in the SUBMIT command.

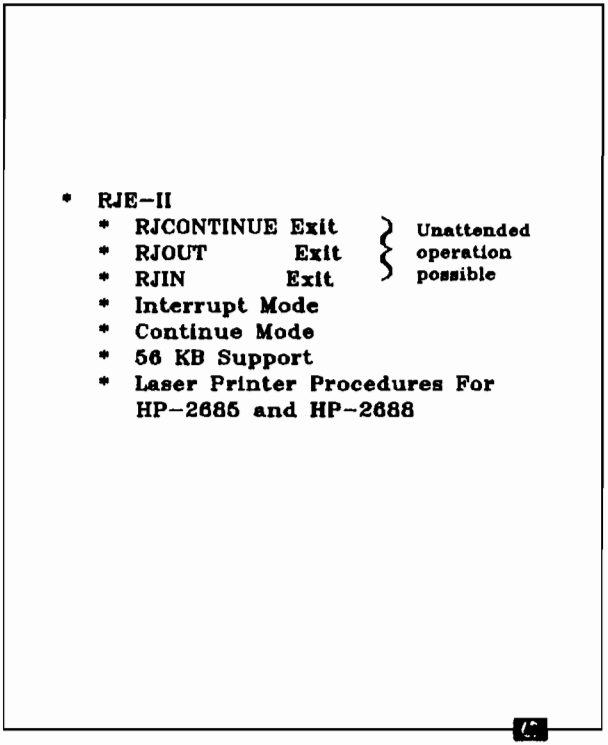
Example 5 lists several of the most complicated batch jobs in terms of output that are submitted by our site. By comparing these against the Remote Terminal and Host configurations, one sees that the output from these jobs can be handled in a desirable fashion and without any intervention on the part of the operator. The JCL listings are deferred for later purging if they are not needed, the forms files are deferred or matched to the appropriate environment files and printers, multipart paper output is labeled for the operator's attention and deferred, and large reports are given a lower output priority.

While no efforts were made to track the costs before and after our new configuration went into effect, some of the savings have been obvious. Just under 10% of our paper usage is saved by not printing the JCL listings, three cases a year on one production job alone. A corresponding savings has been made in phone line costs by reducing the MSGLEVEL such that each JCL listing is now a maximum of 2 pages, as compared to up to 35 pages previously. Multiple copies of reports are no longer sent, and requests for assistance to recreate and retransmit output that had been lost due to mistakes have been almost completely eliminated. Most importantly the operator no longer has to release, alter, copy, print, and otherwise handle every output file that comes from the host. Furthermore the training and experience required for a new operator is greatly reduced.

Conclusion

DSN/MRJE provides a significant amount of flexibility in terms of being able to manage output from the host system, as well as accomidate various hosts and communications facilities. Each site will have to examine their range of output to determine the unique configuration that will best suit their needs. Hopefully this paper has provided a clearer insight into some of the capabilities that are available. An efficient hands off setup for DSN/MRJE is possible and is well worth the effort.

HEWLETT PACKARD

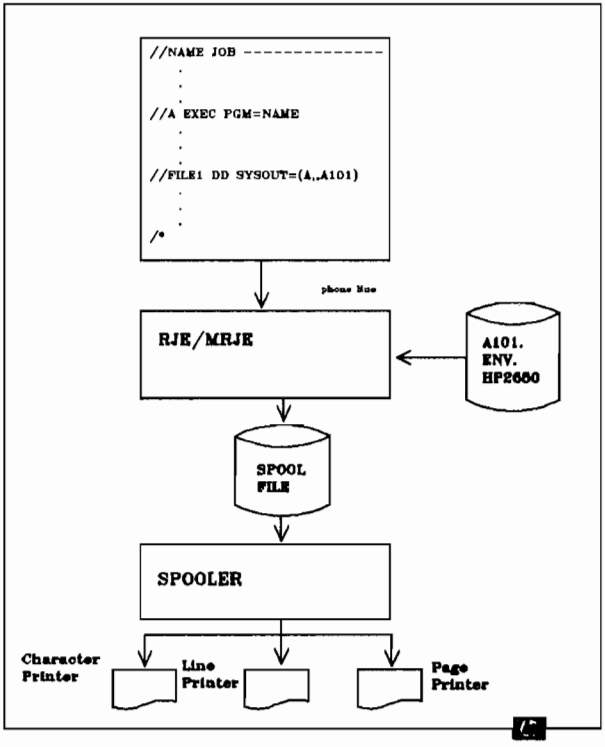


I want to thank you for the opportunity to speak to you today about the status and performance of the HP-3000's datacomm products used to talk with IBM computers.

During 1983 we saw a major release of RJE in which a programmatic user interface was introduced. It is now possible for you to write three different types of user procedures that make RJE significantly more friendly. RJE was also certified to run at 56 KB. HP's Boise division took advantage of the two RJE enhancements and has developed a set of Laser Printer support software to print large volumes of output when the host supports the 3780 protocol (as opposed to HASP protocol).

The introduction of the 'Bonzai' laser printers (HP-2687/HP-2688) brings to three the total of laser based printers that can mix text, form or graphics together on one page based upon the IBM JCL selection of a form name. You now have an $11,000 or $30,000 or $90,000 laser printer solution based upon functional need and print volume.

HEWLETT PACKARD



Both RJE and MRJE allow you to specify the name of a laser printer environment file as follows:

```
//REPORT  DD SYSOUT=(A,,A101)
```

where A101 is the name of a form. On the HP3000, a disk file named A101.ENV.HP2680, is used to print the host output. Since the HP 2680 and 2688 are spooled (only) devices, the host output does not start to print until the last line of output is received. While this at first seems to be a detriment, it can be turned into a significant advantage in most environments.

Since RJE and MRJE both spool, it is possible to specify the number of copies to be printed. Thus, it only becomes necessary to tie up the datacomm line long enough to transmit a report once. Whether two or fifty copies are needed, you only need to transmit it once. The Boise Division RJE exits allow a spool file to be closed and reopened every 'N' pages, as specified by the user.

HEWLETT PACKARD

In 1983

MRJE - Version 2

(1) 56 KB Support (As of A.02.00)
(2) Runs as a batch job (Eliminate 6 MPE SF's)
(3) User modifiable message catalog
(4) Laser printer support to associate an environment file to a formname
(5) Expanded Error Retry

In 1984

(1) Laser Printer support:
- **Based on formname,**
  specify environment file
  number of copies
  spool file priority
  physical form
- **Periodic close/open of spool.files**

(2) Other enchancements based upon user enhancement requests submitted as SMR's

MRJE also received a major product release in 1983. MRJE is no longer part of MPE. MRJE now runs as a batch job. This uncoupling of MRJE from MPE has eliminated six possible MPE software system failures. Significant laser printer enhancements were put into MRJE, and it was certified for 56KB operation. Yet another round of major laser printer enhancements will be introduced in our fiscal 1984 year.

MRJE is now running at a number of customer sites at 56 KB, in order to drive two laser printers at full speed. It has been found that in most environments where multiple part forms are printed, one 56KB line can keep two laser printers busy.

HEWLETT PACKARD

In 1983

- **IMF Pass-Thru II**
  - **Seven Function Keys**
  - **Repaint from "Point of Change"**
  - **Modified Data Tag on 2624B**
  - **Faster Response Time**

- **IMAS/3000 Introduced by Fjerndata**
  - **Non Hewlett-Packard CRT's**
  - **Excellent logging features**
  - **Superior response time**

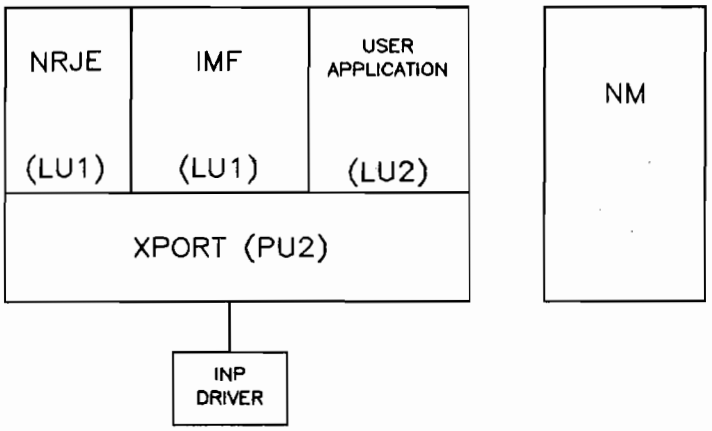
A significant release of IMF was also made in 1983. Q-Delta 2 MIT contained 'Pass-Thru II'. You can now assign IBM PF (Program Function) key names directly to seven of the HP CRT's function keys. The iritating repaint of the same screen contents as new text was received and added to a screen has been eliminated. The response time has been improved.

You may not know that IMF now supports both the SDLC and Bi-sync protocols for the 327x/328x family.

A Norwegian firm name Fjerndata introduced a very well thought out replacement for IMF 'pass thru'. This product is named IMAS/3000. It is not my intention to feature a non-HP product in this talk, but such a product is worthy of mention. You must first buy IMF/3000 and then add the IMAS/3000. Both pass thru products can run concurrently.

HEWLETT PACKARD

RELATIONSHIP OF HP'S FOUR IBM SNA DATACOMM PRODUCTS



NRJE 8100 equivalent implementation of JES2 SDLC batch station

IMF 3270 SDLC Emulation

NM Network Management Software

XPORT 8100 equivalent to Data

For years you have been complaining that HP has not properly supported IBM's SNA architecture. In 1984 HP will be making several major product introductions to add SNA PU2/LU2 support to the HP-3000 family. First we will introduce a batch job entry and print station product that we have chosen to name NRJE, or Network Remote Job Entry. You need the product software of the same name from IBM in order to run the HP software.

NRJE can run on the same datacomm line to the IBM host as does SDLC IMF. You can have both a batch and interactive interface to your host on a single phone line. The 'catch' is that you will need two INP's and a modem sharing device in order to concurrently run both products.

We recognize that it is both awkward and expensive to run two INP's connected to the same host over the same phone line. HP knows you would rather have just one INP and still be able to run both IMF as well as NRJE. So, the second product (which will be separate from NRJE) we call our "SNA Transport" software. SNA Transport is like a multiplexor that allows multiple PU's to concurrently run on the one INP.

HEWLETT PACKARD

IBM Compatible CRT's

2628 adds HPWORD capability to HP-2625

Both CRT's have:
- **IBM Remote Cluster Controller Interface**
- **Two display memories with "mode" toggle button**
- **Dual session capability**
- **Graphics option on ASCII port (HP-2623)**
- **New "Smooth" Scrolling**
- **Optional DEC compatibility**

Both CRT's can replace remote site IBM CRT's and coexist with each other on the same string of terminals

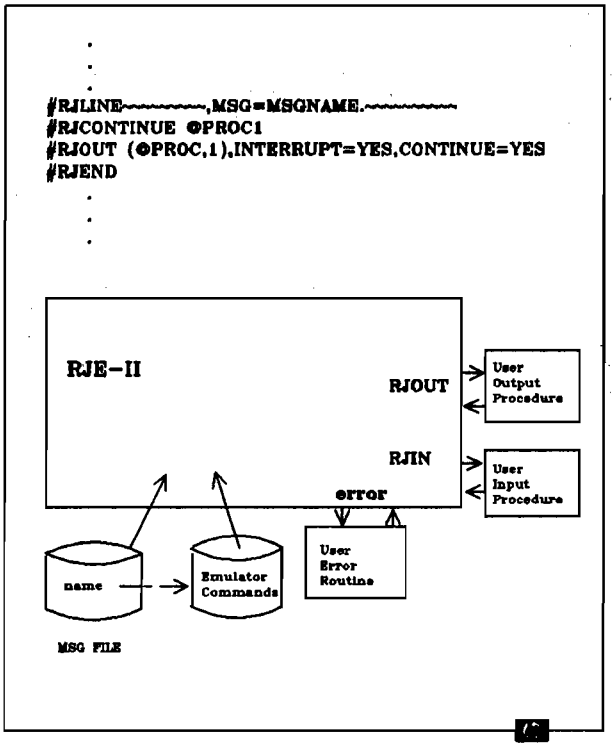
In 1983 HP also introduced two very exciting CRT's of great interest to our customers who wish to communicate both with their local HP-3000 as well as with an IBM host. Both the HP-2625 and HP-2628 are equipped with an IBM-3274 CRT interface on CRT port two (2).

The bottom left hand buttom on the keyboard allows the operator to switch back and forth from ASCII mode to 3270 mode. Two separate display memories allow the operator to truly run simultaneous sessions on two computers. While you are functioning on one CPU, any output from the other modifies the appropriate display memory. When you flip back to the other display space from the current space, you will see any new output delivered while you were functioning in the other display space.

There is now no longer any reason to tie up a large amount of desk space with both an ASCII and 3270 CRT sitting side by side.

Both of these new CRT's allow graphics to be added, including Tektronix compatibility mode. In addition, the HP-2628 can be used with HPWORD. The HP-2628 can be downloaded by HPWORD to function just like an HP-2626W CRT. Both CRT's support the DEC compatibility option on their ASCII interface.

HEWLETT PACKARD



RJE has been given a non-stop default to be waiting for host output any time it is not busy submitting a batch job. The default mode is interruptible any time RJE is idle, to allow a job input, and then goes back to waiting for host output.

This slide shows how RJE-II uses the concepts of INTERRUPT and CONTINUE to setup the default RJOUT. You stream RJE as a batch job consisting of four emulator commands: RJLINE, RJCONTINUE, RJOUT and RJEXIT.

The RJLINE command establishes the host link. The RJCONTINUE command names a user procedure to be given control anytime RJE encounters an error. Without the RJCONTINUE, the RJE program might terminate prematurely. The RJLINE command tells RJE the name of a MSG file to read for the names of other disk files with emulator commands. This is the INTERRUPT queue. The RJLINE procedure "chops up" the continuous host stream into individual output files.

HEWLETT PACKARD

It is the INTERRUPT=YES that causes RJE to keep a read pending on a MSG file. Whenever RJE is idle, it is reading the MSG file. When a read completes, RJE expects to have read the name of yet another MPE file in which it will find RJE emulator commands. Once the set of commands are handled, it is the CONTINUE=YES command that causes RJE to resume the RJOUT that was interrupted.

The first INTERRUPT for RJE is usually to go thru the expected host signon procedure. So, the batch job needs to have placed the SIGNON commands in a disk file, and then placed the disk file name in the MSG file, and then RUN RJE with our four emulator commands.

As soon as RJE goes into execution it will open the host interface and find no output coming across because the host is still waiting for RJE to SIGNON. So, with no output from the host, RJE will read the MSG file. The MSG file will point to the file with the SIGNON commands, which the emulator will obey. Once signed on, the host may start sending output that the original RJOUT will handle.

HEWLETT PACKARD

The HP-150 Will Emulate A 3270

An option will be introduced in 1984 to plug the HP-150 Personal Computer directly into an IBM-327x cluster controller using an IBM coaxial cable. As such the HP-150 will work on both local and remote cluster controllers.

Because the HP-150 uses a coaxial interface to an IBM supplied (or compatible) cluster controller, they are independant of the datacomm protocol used. This makes the HP-150 particularly well suited to upload and download IBM host data to PC disk.

HEWLETT PACKARD

NOTES:

CONTROLLED DATA COMMUNICATIONS INTERFACING WITH REMOTE MICRO COMPUTERS

Preface

"If you can't dazzle them with brilliance, baffle them with something else!" most aptly applies to the title of this paper! What is controlled data communications? How does that apply to an interface? How is a remote micro computer defined?

How do you control datacom and what is it when you get it controlled? Data Communications is a means of transporting data thru the most unreliable and unpredictable means, generally a phone line, for use by the recipient. You control it by establishing reasonable precautions and checks at each end of the communication to affect a 97% (plus) reliability factor. Expect anything and everything when building in your precautions and you will not be disappointed. This paper will show how we did what we did and may give you some ideas for your situation. Beyond that, attend the presentation and ask your questions. As to what you have when you attain control over datacom, it turns out you end up with a decent tool to use to meet today's data processing requirements.

How does controlled datacom apply to an interface? If you can build a controlled datacom network --a reliable one-- you can use it to allow equipment to communicate thereby, easing a manual task or being more effective. If what you are doing is not ending in a more efficient and effective situation, don't do it.

For our purposes, a remote micro computer is described as a micro with limited intelligence situated in an environmentally unstable area, away from trained data processing personnel but accessable via a reliable datacom network. Quite a feat if you can accomplish it.

Section I - Introduction & General Description

E-Z SERVE, INC. AND SUBSIDIARIES

E-Z Serve, a petroleum refining and marketing company whose home offices are located in Abilene, Texas, has retail outlets thru most of the "sunbelt" states. The company got started in 1971 and utilized outside D/P services until 1980 when we purchased our first Hewlett-Packard computer: a series III. Since 1980 we have purchased two more series III's and a series 64 to accomodate our general data processing requirements. The original series III is used for system development and backup in that new software is created on this system and transported to other systems for production.

In 1981, E-Z Serve entered the micro world by virtue of Automated Fueling Systems. At that time few companies developed such equipment and fewer still understood the need for true flexibility in the design of their software. As a result, we developed a system of our own utilizing the Motorola 6809 CPU chip as a base. In order to do this we purchased a Hewlett-Packard 64000 Logic Development System to develop the software and aid in debugging and system analysis.

THE HEWLETT-PACKARD 3000

Our system design called for the HP 3000 to be utilized as a tool in communicating with our Fuel Site Micros. This provided quite a challange as our objectives called for the use of standard products only (see Section II - Objectives). With a fair amount of research we found that this was possible providing some minor allowances were made within the micro to allow flexibility in the data communications area. In doing the research we found that

within the Data Processing industry, Data Communications is the newest and most underspecialized field around today. I strongly recommend the Hewlett-Packard course entitled "Introduction to Data Communications" for anyone doing even a medium amount of work with datacom. Those of my people who have attended this course have shown a much greater understanding of the field upon completion of the course.

THE FUEL SITE MICRO

The Fuel Site Micro is a Motorola 6809 based microcomputer. This unit, located at the fuel site, handles the communication necessary to authorize purchases requested by a cardholder with an acceptable magnetically encoded debit or credit card. Control information is downloaded to the micro and stored transactions are uploaded to the host as a part of the interaction between the micro and the HP 3000. This interaction, what we learned and the sources available for information, make up the subject matter for this paper.

Section II - Objectives.

Our objectives were quite simple. We wanted to create a situation which required little or no manual intervention yet allowed us to control micros, as required, from a host computer. This host was to remain as standard as possible with respect of functioning within manufacturer's unaltered specifications and utilizing generally accepted data processing techniques and procedures. Peripheral equipment was to be utilized in an off-the-shelf fashion and again was to be a device generally accepted to the data processing community in a non-customized state. We recognized that customization cost more and wanted to avoid this unnecessary expense.

The majority of the code was to be done in an easily maintainable language with general subroutines for special functions written in a more detailed language (in this case SPL). The use of intrinsics was to be the exception rather than the rule. We wanted to utilize readily available software tools already on the market to relieve programming effort where possible. This was done thru some of the Quasar products.

Another important factor was data security. This was addressed thru the use of data encryption routines. A sample DES (Data Encryption Standards) routine is included in the Appendix. References are made to other sources available for this information. We also created our own encryption process and my recommendation to anyone is to do the same just for an added measure of protection. Your situation will be different and I recommend you expound these objectives for your use.

Relatively important to us was the issue of backup support. With our multiple HP 3000 processors we are able to duplicate software and data files at separate facilities in order to allow for continuous support in the event of system failures. We utilize DS/3000 to create and maintain the support files necessary at each site. This has allowed us to switch support from one facility to another easily and we can switch back to the original facility just as easily. Because we stipulated the use of standard features we can carry backup tapes to virtually any HP 3000 site with an autodialer modem and be in position to support the micros within 1 hour of arrival. This is important due to the need to provide reliability to the user of the micro.

It is easy to become enchanted with all the nice features available on equipment today. My best advice is to See It Big and Keep It Super Simple (SIBKISS). Keep the overall objective in mind but work at the simplest level possible on each step to that objective.

Section III - Line Control Techniques

Communications with the Fuel Site Micros is done with dial-up, asynchronous phone lines operating at 1200 baud and even parity. The relationship between the HP 3000 and the Fuel Site Micro is such that the HP acts as the master, initiating and directing the data transfer between the systems. The host computer interacts with the micro using ASCII commands and responses. Commands sent to the micro consist of a 'DC3', a two character command and any neccessary parameters followed by a 'DC1'. Since the FREAD intrinsic prompts with a 'DC1' only the characters up to the 'DC1' are written to the device in order to allow the HP to be ready for the response. In the case of multi-record responses the pacing

of each record is accomplished by the micro waiting for a 'null' command, consisting of a 'DC3' and FREAD's 'DC1', before sending the next record. In some cases, this transfer must be interrupted. If this interruption is to come from the host, a simple command is sent to the micro. Usually, an interruption will not be caused by the micro unless the phone connection is lost or the micro system goes down. In this event, the host programs recognize that fact and adjust themselves accordingly. Most of the data transfer is handled in this way since data is primarily sent from the micro to the host. In some instances we must download a large list of numbers to the micro. Transmitting this list can take as long as 2 hours. Since the characteristics of these numbers allows this list to be stored in a bit map, it could also be transmitted in this form. The bit map is sent in blocks of 2048 numbers as follows:

HP: 'DC3' CCnn 'DC1' where 'CC' is a two character command and 'nn' is the block number

micro: D or ? response indicating whether successful or not

HP: 'DC3' 'STX' hhh ... hhh 'ETX' cccc 'DC1' where 'h' is one of 512 hex digits, each one representing a nibble, and 'cccc' is the check digit

micro: D or ? response

This technique transmits the entire list in about 3 to 5 minutes or approximately 3% of the original time required.

During any typical access of the micros, the port is opened and closed several times as the host program calls each location. A method was needed to prevent multiple processes from using this line. One way would be to simply try to open the port; if the open is successful, proceed, if not, terminate the program and try again later. This approach, however, would not work in this case since, as mentioned before, the port would be opened and closed many times during one access and one program could sneak in and open the line while the first program is between phone calls. A more practical method is to use global Resource Identification Numbers (RIN's). Each program that needs the phone line, first trys to lock the RIN, then continues with all of its various line opens and closes, and, once it's finished, unlocks the RIN, freeing up these resources for use by other processes.

Section IV - Use of Hayes Smartmodem 1200

In our application it was neccessary to have requlary scheduled, automatically initiated communication with the Fuel Site Micros. This involved the use of a sleeper job on the 3000 with some kind of auto-dialer to make the calls. After reading a product review in Byte magazine, we decided to try out a Hayes Smartmodem 1200. The Hayes Smartmodem 1200 works with standard RS-232C in full or half-duplex at 1200 or 0-300 baud. It is compatible with Bell 103 and 212A modems in asynchronous mode only. Since this unit is a self-contained, stand-alone system, it required no additional equipment for use other than the computer cable. Connections with phone lines is with a standard modular telephone plug (RJ11, RJ12 or RJ13). One of the really nice features of this modem is the audio monitor included within the cabinet that can be very useful when trying to determine why a call is not being completed. On the front of the enclosure are the usual LED indicators (carrier detect, receive data, modem ready, etc.) with additional ones for auto-answer mode, and off-hook condition. The excellent documentation provided with the modem along with its reasonable price ($500 to $700) made this product very attractive to us.

Controlling the Hayes is accomplished by commands sent to it over the RS-232 line and an 8 position DIP switch. The modem provides a full compliment of simple ASCII commands with responses in English words or decimal digits. For instance, there are commands to turn the speaker on or off, control echo and half/full duplex, select default dialing modes (pulse or tone), set various configuration register values, and of course dial a phone number. The responses include 'CONNECT', 'NO CARRIER' and 'ERROR' or their numeric equivalents: 1, 3 and 4 (the selection of words or digits is also by a command.)

Using the Hayes with the 3000 is generally quite simple. Rather than requiring an explicit command to hang up the line you can choose to let the HP's dropping of RS-232C DTR signal instruct the modem to hang up the phone when the port is FCLOSEd.

Although the word responses are more explict and less likely to be confused with data, it is neccessary to use the digit responses because of the way the word codes are returned. Each word response includes a carriage return character both in front of and behind the word. This leading carriage return terminates the read of the response and the HP hasn't got

a chance of being ready for another read to accept the rest. The digit responses, however, have a carriage return only after the digit, making them much easier to be read by the HP. Another related item is that the Hayes does not wait for a 'DC1' to send a reply, so if you expect to receive any response you must give it something to do to keep it busy while the 3000 is getting ready for the read. In other words, any configuration commands can and should be included in the same command string as the dial command to give the HP enough time to receive the response.

Section V - Software Tool Development

Before writing application programs, a few tools had to be developed first, mainly subprograms to take care of the actual interface with the micro since most application programs would be written in COBOL. While developing the subroutines, we found a lot of useful information in Ross Scroggs and John Tibbets' articles in the 1982 San Antonio proceedings. Subroutines were written to open and close the line, read from and write to the micro, dial the phone, and logon to the Fuel Site Micro system. The following is a discussion, along with examples, of some of the subroutines developed.

FSMOPEN

The FSMOPEN procedure opens the port and sets up a few line characteristics. To provide flexibilty of use between our different systems and configurations, we used a device class name when opening the port rather than a device number since different systems had device classes assigned to different ports. Also to distinquish between our auto-dial port and our standard dial up port, we gave the auto-dial LDEV an additional device class name (AUTODIAL) along with the common name (DIALUP) (See the configuration listing in the appendix).

```
MOVE FILE'NAME := "FSMFILE ";
MOVE DEV'CLASS := "AUTODIAL ";
FSM'FILE := FOPEN (FILE'NAME,%604,%104,-540,DEV'CLASS);
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 GO TO EXIT;
 END;
```

The only other significant item to point out here is that the port is opened with CCTL in order to use the %320 carriage control parameter in writing to the FSM since the micro does not use carriage returns to delimit commands.

After the port is opened, we must set up the term type, baud rate, parity and echo attributes. A sample of these might be:

```
PARAM := %7411;                    << SET TERM TYPE = 9 & >>
FCONTROL (FSM'FILE,37,PARAM);      << BAUD RATE = 1200    >>
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 GO TO EXIT;
 END;

PARAM := 2;
FCONTROL (FSM'FILE,36,PARAM);      << SET EVEN PARITY >>
IF < THEN ...

FCONTROL (FSM'FILE,24,PARAM);      << ENABLE PARITY CHECKING >>
IF < THEN ...

FCONTROL (FSM'FILE,13,PARAM);      << TURN ECHO OFF >>
IF < THEN ...

FSETMODE (FSM'FILE,4);             << SUPPRESS CR/LF >>
IF < THEN ...                      << AFTER READ     >>
```

FSMCLOSE

All this procedure does, other than closing the line, is pause for the five seconds during which DTR is down after the file is closed before proceeding to the next phone call.

```
FCLOSE (FSM'FILE,0,0);
IF < THEN ...

INTERVAL := 5.0;
PAUSE (INTERVAL);
```

FSMREAD

When reading a programmatically controlled device you should take precautions to prevent your read from hanging, should the device fail to respond. The easiest, if not the only, way to do this is with timed reads.

```
TIME'LIMIT := 2;
FCONTROL (FSM'FILE,4,TIME'LIMIT);
IF < THEN ...

LEN := FREAD (FSM'FILE,FSM'BUF,-540);
IF > THEN
BEGIN
 FSM'STAT := 1000;       << INDICATES EOF TO CALLING PROGRAM >>
 GO TO EXIT;
 END;
IF < THEN ...
```

If the FREAD does time out, the calling program will be given the file system error (22) to handle as it needs to.

FSMWRITE

The calling program will pass to FSMWRITE the command minus the leading and trailing delimiters. The trailing 'DC1' comes from the following FREAD but the leading 'DC3' is inserted by this procedure before transmission. Along with the buffer to be sent, the calling program passes a length which follows the same rules as the length parameter in FWRITE intrinsic (positive -- word count; negative -- byte count).

```
IF FSM'LEN < 0 THEN      << CONVERT TO POSITIVE BYTE COUNT >>
 LEN := \FSM'LEN\
ELSE
 LEN := FSM'LEN * 2;

MOVE B'BUF := %23,2;
MOVE *     := B'FSM'BUF,(LEN);

FWRITE (FSM'FILE,FSM'BUF,-(LEN+1),%320);
IF <> THEN ...
```

FSMDIAL

The procedure to dial the Fuel Site Micro using the Hayes Smartmodem 1200 accepts a phone number string from the calling program. Setting the modem for half-duplex and decimal digit responses as required can be done with commands but we chose to use the internal DIP switches since this configuration would always be used. The phone numbers used for each location called are stored in a controlling file in the exact format as will be passed to the dialing procedure including 1 + area code for any long distance calls. The string 'AT T D' does several things: 'AT' is the character sequence telling the Hayes that this is a command for it to execute, the 'T' instructs the Hayes to default to tone dialing mode, and the 'D' begins the actual dial command. If the number must be dialed using pulses, a 'P' can be inserted in front of the number stored in the control file. Examples of phone number dialing sequences are:

```
AT T D 555-1234

AT T D 1 (808) 555-1234
```

```
AT T D P 555-1234
```

Spaces, parentheses and hyphens passed to the Hayes are ignored and can be added to improve readabilty. The procedure will set up the phone number dialing sequence, pass it to the Hayes, and return a response indicating results.

```
FRWRITE (FSM'FILE,L'BUF,0,0);      << SEND INITIAL CR >>

MOVE B'BUF := 256 (" ");           << CLEAR BUFFER >>
MOVE B'BUF := "AT T D ",2;         << BUILD COMMAND >>
MOVE *     := B'FSM'PHNUM,(20);

LEN := 28;
DO LEN := LEN - 1                  << FIND LENGTH OF          >>
 UNTIL LEN < 6                     << SIGNIFICANT CHARACTERS >>
 OR B'BUF (LEN) <> " ";
IF LEN = 6 THEN                    << NO PHONE # PASSED >>
BEGIN
 FSM'STAT := 1001;
 GO TO EXIT;
 END;

FWRITE (FSM'FILE,L'BUF,-(LEN+1),0);

MOVE B'BUF := 256 (" ");

TIME'LIMIT := 30;
FCONTROL (FSM'FILE,4,TIME'LIMIT);

LEN := FREAD (FSM'FILE,L'BUF,-256);    << GET RESPONSE >>
IF > THEN
BEGIN
 FSM'STAT := 1000;                 << INDICATES EOF TO  >>
 GO TO EXIT;                       << CALLING PROGRAM   >>
 END;
IF < THEN
BEGIN
 FCHECK (FSM'FILE,FSM'STAT);
 IF FSM'STAT = 22 THEN
  FSM'STAT := 1002;
 GO TO EXIT;
 END;

IF B'BUF <> "1" THEN               << DIALING FAILED >>
BEGIN
 FSM'STAT := 1002;
 GO TO EXIT;
 END;

FSM'STAT := 0;
```

FSMLOGON

In order to secure the Fuel Site Micro from unauthorized access, a logon procedure was established which included a password stored with the control file for each location. The procedure will not be shown here for security reasons but the actual code did not involve anything complicated.

After the subroutines had been developed and testing began a program was needed that would emulate the Fuel Site Micro's system console. This program's initial purpose was only to assist in testing of other programs using the communications interface to the micro. This program later proved to be very useful for both development of new application programs and for setup and maintenance of the Fuel Site Micros. As new programs were needed to load or collect a particular set of data, this program, along with a data line monitor, could be used to observe exactly how this was to take place, including such things as errors that can occur, responses, terminating the transfer, etc. Also, as stations were brought up or reconfigured, the console emulator could be used to download

a command file of setup parameters and on a day to day basis, monitor and maintain these parameters from the host as if the operator were at that location.

As mentioned earlier, a sleeper program was needed to initiate the data collection program. At first, we wrote a quick and dirty sleeper that simply streamed a job and went to sleep for a specified time interval. This program worked well enough, but we later decided that it would be necessary to automatically stream a few other jobs at different time intervals than that of the data collection job. After checking the contributed library catalog, we found the SLEEPER programs. These programs consists of a sleeper that actually performs the functions and a maintenance program that updates the control file. The use of a control file makes this system very flexible and easy to use as it can stream jobs, run programs, execute MPE commands and files of commands. Our testing and use of these programs showed them to be very useful and flexible.

Section VI - Error Checking

Although there is a certain amount of error detection and correction built into the modems, there are still many transmission errors that can occur. With the addition of a few simple procedures, error detection, and a certain degree of correction, could be greatly increased. The primary method of doing this was by making use of check sums. Almost all data transfer that occurs automatically is numeric, consisting of either decimal or hexidecimal digits, and includes a check sum that is computed and compared by the host. Another area of checking was passed to MPE itself by enabling parity checking. If a parity error or check sum error occurs, the host requests that same record from the micro, up to three times, until no errors occur. Also, since there is a possiblity of missing records or collecting the same records twice due to an abnormal previous collection, each data record collected is assigned a request or transaction number by the micro. The HP keeps track of which records were received as of the last collection and reports any discrepancies (missing or duplicated records) which may require manual intervention. These few simple procedures greatly enhance those built in and provide an adequate amount of error checking.

Section VII - Remote System Backup

We have several HP 3000's in various locations connected via DS lines so we decided to design some features of the system in such a way as to make remote system backup as easy as possible. First of all, each of our four systems has at least one dial-up port configured as device class 'AUTODIAL' (See configuration listing in appendix) with a Hayes Smartmodem 1200 attached. Also, all software and at least a skeleton data base is duplicated at each of the sites. The controlling of the automatic communication is done, in part, by a location file. Each of our locations for each of our systems has an entry in this file with flags indicating such things as whether or not to call, what data to collect or down load, etc. By controlling our programs from this file, along with a minimal amount of other data in the data base, it became quite simple to switch data communication tasks from one system to another. To make the switch usually involves nothing more than changing a few flags, and adding a new task to the sleeper control file.

Summary

It is possible to affect a reasonably reliable data communications technique without doing anything exotic in the hardware or software.

This paper contains a very simple and basic approach which we found acceptable. All sloppy procedures and inefficient coding are probably oversights and should be treated as such. No one is trying to state that the indicated techniques are the best or only way of accomplishing the end results. If better is possible, good is not good enough! So by all means improve on this information in any way you can.

BEST OF LUCK TO ALL OF YA'LL!

DISCLAIMER

The ideas and material presented in this paper have worked for us but it is entirely the reader's responsibility to assure that any material utilized by the reader is functional for the reader's intended purpose. The authors disclaim any and all responsibility or liability which may arise directly or indirectly from the use in whole or in part of any of the ideas, concepts, examples or other material presented herein. The authors are presenting this paper as individuals seperate and apart from any company affiliation.

ACKNOWLEDGEMENTS

This paper was co-authored by Mike Shelton. I would like to express my appreciation to Mike not only for his assistance on this paper but also for doing a superb job in assisting in creating a CONTROLLED DATA COMMUNICATION INTERFACE TO REMOTE MICRO COMPUTERS for our shop.

Other acknowledgements are as follows:

Hewlett/Packard	for superb development gear
Ross Scroggs	for related papers and work
Chris Gindorf	for professional help and direction
Tommy Sorrells	for efforts beyond the call
Judi Pianta	for beginning footsteps
Sharon Ashby	for being part of my life
Randy Nicholson	for the challenge and opportunity

GLOSSARY OF SELECTED TERMS

Bit Map: a representation of data where a particular bit within a byte has positional value rather than numerical value; the on/off status of a bit represents a condition; examples: bit mapped graphics where each bit represents the on/off status of a pixel on the screen; table of numbers where each bit represents the absence or presence of a number by its being off or on.

CCTL: Carriage control

Check Sum: a sum of the numerical values of each character in a particular buffer.

DC1: ASCII character; Device Control 1; decimal equivalent: 17.

DC3: ASCII character; Device Control 3; decimal equivalent: 19.

DTR: Data Terminal Ready; pin 20 of RS-232C interface.

ETX: ASCII character; End of Text; decimal equivalent: 3.

FCLOSE: HP system intrinsic or subroutine used to close a file.

FREAD: HP system intrinsic used to read from a file.

STX: ASCII character; Start of Text; decimal equivalent: 2.

REFERENCES

Computer Data Protection, Federal Register, Commerce Department/ National Bureau of Standards, March 17, 1975, pp. 12134-12139.

McEntire, Norman C. "Hayes's Stack Smartmodem." Byte, March 1983, pp. 282-290.

Meushaw, Robert V. "The Standard Data Encryption Algorithm, Part 1: An Overview." Byte, March 1979, pp. 66-74.

Meushaw, Robert V. "The Standard Data Encryption Algorithm, Part 2: Implementing the Algorithm." Byte, April 1979, pp. 110-130.

MPE Intrinsics Manual, Hewlett-Packard Company, January 1981

Scroggs, Ross. "Everything You Wanted to Know About Interfacing to the HP3000, Part I.", Proceedings of the 1982 HP 3000 IUG San Antonio Conference, February 1982, pp. 6-40-1 to 6-40-10.

Smartmodem 1200 Owner's Manual, Hayes Microcomputer Products, Inc., 1982.

Tibbets, John J. "Everything You Wanted to Know About Interfacing to the HP3000, Part II.", Proceedings of the 1982 HP 3000 IUG San Antonio Conference, February 1982, pp. 6-40-11 to 6-40-17.

LOG DEV NUM	DRT NUM	U N I T	C H A N	T Y P E	SUB TYPE	TERM TYPE (DEV	SPEED TYPE)	REC WIDTH	OUTPUT DEV	MODE	DRIVER NAME	DEVICE CLASSES
1	25	0	0	3	8	(7935)		128	0		HIOMDSC2	DISC SPOOL SYSDISC
.												
.												
.												
52	8	32	0	16	0	10	240	40	52	JAID	HIOTERM1	TERM
53	8	33	0	16	1	10	240	40	53	JAID	HIOTERM1	AUTODIAL DIALUP
54	8	34	0	16	1	10	240	40	54	JAID	HIOTERM1	DIALUP
55	8	35	0	16	0	10	240	40	55	JAID	HIOTERM1	MUX

DAVID L. ASHBY is a Texas native with 10 years data processing experience. This experience spans banking, college, service bureau, and petroleum industries with a good mix of applications within each.

David is currently Data Processing Manager for E-Z Serve in Abilene, Texas and has been in that job since early in 1980. Among the unique experience is his overseeing and developing of an Automated Fueling System on Micro Computers. These units are interfaced into HP 3000 computers -- thus the base for this paper and associated presentation.

---

1 9 8 4
HP3000IUG
ANAHEIM
CONFERENCE

PRESENTATION ABSTRACT

Title: Portable Development Between the HP 3000 and a Microcomputer Using Pascal.

Author: Kenneth C. Butler
Manager, Technology
Imacs Systems Corporation

Abstract: Corportations, which are now facing the appearance of microcomputers in their user community, are often discovering that many requests for microcomputer applications require custom programming. This usually means hiring contractors and accepting systems written in languages specific to one microcomputer. Supporting users with such systems becomes problematical when the in-house data processing staff has no expertise in the language chosen, and when data adminstration and auditing staffs attempt to verify the proper use of corporate assets. With Pascal/3000 it is possible to develope portable applications between the HP 3000 and microcomputers which use one of the common dialects of Pascal. Using features of Pascal/3000 to construct a carefully chosen subset of extensions found in microcomputer Pascals, programs can be written which compile on either the HP 3000 or microcomputer with few, if any, changes. It is also possible, in some microcomputer Pascals, to construct intrinsics that mimic many common MPE intrinsics. Finally, some microcomputers support environments uncannily close to a "single tasking MPE" environment; i.e. 64Kb separate code and data stacks; a hierarchical file system whiuch can simulate "groups" and "accounts"; and program and system code libraries.

1 9 8 4
HP3000IUG
ANAHEIM
CONFERENCE

Portable Development Between the HP 3000 and a Microcomputer Using Pascal.

Kenneth C. Butler
Imacs Systems Corporation

1. Introduction

For many corporations the much heralded microcomputer revolution is finally arriving. One way this happens is the gradual acquisition of personal desktop computers by key departmental managers, who intended to use "primary" off the shelf applications, like spread sheet and word processing programs. As these applications prove successful, more microcomputers are purchased, are given to the professional and secretarial level persons throughout a department, and more off the shelf applications, like data base managers, come into use. Somewhere during this process requests begin to surface for applications which demand just a little bit more than the off the shelf programs can provide. These requests may involve applications such as:

- Extracting information from files and data bases from one computer (usually the HP 3000 host) and transmitting it to another computer (usually the microcomputer). Typically this is first encountered with spread sheet users needing current corporate data.

- Handling data in the unique way a corporation does business and which may not be easy to implement using a standard package. For example, a simple data base of international sales orders might require extensive currency conversion based upon the specific date of each transaction rather than the overall balance of an account.

- Installing a complex, but small capacity, data processing system for some service department. An example of this might be an inventory control system for a stock room; this system might include vendor information for reordering stock and customer information for "billing" other corporate departments for useage of the supplies.

It is at this point that the corporate data processing staff becomes involved, often for the first time. Other than ignoring the situation, which happens in a larger percentage of the cases than you might suppose, the dp staff can use several approaches to dealing with the situation:

- Let the users become programmers and develop their own applications.

- Hire contractors who are experts in the various microcomputer systems to do the development.

Both of these approaches usually mean accepting systems written in languages or data management systems which are specific to one microcomputer or family of microcomputers. It also usually means accepting a system which has no commonality with the HP environment that the corporation is using. Supporting users with such systems becomes problematical when the in-house data processing staff has no expertise in the language or systems chosen, and when data administration and auditing staffs attempt to verify the proper use of corporate assets.

What is ideally needed is a way for the corporate dp staff to become directly involved, using tools that they have ready access to in the HP environment. Features needed in these tools include:

- A common language should be found on both the HP and the microcomputers, so that a primary or "kernel" system can be developed on the HP and then transported to the microcomputers.
- A high degree of portability should be supported so that the same application can be developed for different target microcomputers.
- The tools on the microcomputers should allow extremely low-level access to the basic file system of the microcomputer and to its hardware, so that portions of an application which must be machine specific may be easily developed, kept isolated, and have a standard "interface" with the kernel application.

To meet these requirements we might try to use a highly flexible DBMS package, like dBase II, or Condor, but the problem here is no HP 3000 support for these types of microcomputer DBMS. Or we might try using COBOL, which exists in several microcomputer implementations, but the problem here is the general lack of low-level access, which becomes critical when we are dealing with screen handling and data communications. We can rule out BASIC because of the highly specific implementations found on each computer. In fact, the one common language which seems to fit best is Pascal.

This paper will examine HP Pascal/3000 and several common microcomputer Pascal dialects to identify areas of similarities and differences. The microcomputer dialects to be covered are:

- Apple /// Pascal
- Pascal/MT+ (CPM/80 version)
- IBM PC Pascal

The depth of coverage of the microcomputer dialects will vary, corresponding with my familiarity with them, and the extent that I have actually used them. In the case of Pascal/3000 and Apple /// Pascal, I have used both extensively and have "ported" programs of varying complexity between these two versions. I am not quite as familiar with Pascal/MT+, having only occasionally worked with it. My knowledge of IBM PC Pascal is currently quite light and mostly comes from examining its documentation, but I have attempted to include it because of the wide interest in this dialect.

In comparing these dialects, I will attempt to specify a few general rules for increasing portability in the following areas:

- Data types and packing
- Procedures, functions, and statements

I will also examine the several specific topics which seem to be of concern when developing a portable application:

- String handling
- File access
- Screen handling
- Heap management

Then I will briefly look at constructing intrinsics in the various Pascals to improve the fit of non-standard features. The issue of separate compilation will be addressed here.

Finally, I will look at the basic operating systems that these dialects use to look for common environmental features with MPE.

2. An Overview of the Pascal Dialects

Pascal was devised by Niklaus Wirth in 1968, and was primarily intended for two purposes: to teach fundamentals of programming as a systematic discipline, and to implement programs in a reliable and efficient way. However, the language as originally defined (Jensen & Wirth, 1974) lacked many features needed in a general purpose systems and applications programming tool. These missing features were in the following areas: interactive file operations (i.e. to terminals and communications lines); random access to disc files; the handling of arrays which might vary in size during program execution; and the separate compilation of program segments.

All of the Pascal dialects examined here have extended the language beyond the original Wirth definition to provide for these missing features. While there are at least two major standards organizations (ANSI and ISO) drafting standards for Pascal, and while many of the dialects to be examined are "based" upon one of the proposed standards, all of the Pascal dialects examined here reflect slightly different approaches to the same extensions. In addition, the UCSD Pascal system, which has become a machine independent "standard" of its own, has set the standard for handling strings which many other Pascals follow closely.

HP Pascal/3000

Developed and marketed by HP, this conforms to the proposed ANSI standard with extensions. It is the most extended of the languages covered. Enormous sets, arrays, the return of structured data from functions, are all things to be wary of.

Strings and string intrinsics have been provided which provided relatively close compatibility with the UCSD string implementation.

Low level support of the operating system is provided by MPE intrinsics in the SYSTEM library.

Separate compilation of code is supported and modules may be bound to the program using the SEGMENTER, or may be placed in a library for resolution at run time.

Code and data stacks are separate, with 64Kb available in each. Various kinds of system overhead can shrink the data stack, as with other languages on the HP 3000. Using MPE intrinsics, it is possible to allocate additional data segments and data may be moved into and out of the data segments onto the normal data stack.

Apple /// Pascal

This version of Pascal, like the version found on the Apple // family and to some extent on the Lisa, is based upon UCSD Pascal version 2.0, and is marketed by Apple Computer.

Low level support of the operating system is provided in two ways; through built-in UNIT I/O procedures, and through an external library of SOS I/O intrinsics.

Separate compilation is fully supported by the UNIT concept, which is slightly different than the method found in the other dialects. One major difference is the way data types and variables may be declared in a UNIT and then referenced by the program "using" the UNIT as if they had been declared globally in the program. Separately compiled units may be bound in a separate linking step (if they are "regular" units) or may be placed in a library for resolution at execution time (if they are "intrinsic" units). Program segments may be declared which overlay memory.

Code and data stacks are separately maintained. On an Apple ///, there is a full 64Kb available in each. Certain kinds of system overhead, such as file blocks, shrink the data stack, but buffer space for screen text and graphics are maintained separately from the data stack. By using SOS memory management calls, it is possible to allocate additional data segments and move data into and out of these segments onto the normal data stack.

Pascal/MT+

Developed and marketed by Digital Research, this conforms to the proposed ISO standard with some limitations and extensions.

This has word and byte extensions and also has many bit oriented features. Numerous extensions are designed to increase compatibility with UCSD Pascal. These include the implementation of STRINGs and string intrinsics, which are identical to UCSD, the provision for BLOCK I/O, and UCSD style HEAP management support, which requires two short user written routines.

Low level support of the native operating system is provided through various system library intrinsics. One feature unique to this dialect is the ability to code assembly language directly in the source of the Pascal program by using the INLINE option.

Separate compilation is extensively supported, but separate code segements must be bound by a separate linker program; the concept of an intrinsic library resolved at execution time is not supported. Separate code segments may overlay portions of memory.

Code and data stacks are not separated and come out of the same address space (about 56Kb on a CP/M-80 system).

IBM Personal Computer Pascal

Developed by Micrsoft and marketed by IBM, this conforms to the proposed ISO standard with some limitations and extensions.

This has extensions for word and byte handling that are designed to work "close to the machine". Support for strings is the most different of the dialects examined. There are actually *two* types of strings; STRING, which is not variable in length, and LSTRING, which is variable in length. String intrinsic support is patterned after UCSD, but is not exactly equivalent as some intrinsics work only on LSTRING, and some on a combination of STRING and LSTRING.

Low-level access to the operating system is provided by various library modules, which require linking to the user program before execution.

Separate compilation is also supported, but separate modules must be linked prior to execution.

3. General Notes on Portability

In spite of the differences between the various Pascal dialects, it is relatively easy to devise a portable subset. When specifying a portable subset, a good rule of thumb might seem *If it's an extension, leave it out*, but we shall see that often the extensions to Pascal are highly compatible and the greatest differences actually occur in the fundamental implementations of the language itself. And where extensions are necessary or desirable and do not seem particularly compatible, it is often possible to develop modules in each of the dialects which perform very similarly.

Data Types

The "standard" Pascal data types are BOOLEAN, CHAR, INTEGER, and REAL. Extensions to simple data types found in microcomputer dialects include BYTE, WORD and various ADDRESS types, and STRING. With the exception of STRING, these extended data types should be avoided. In all of the dialects considered, the CHAR data type is functionally equivalent to BYTE when used with the ORD and CHR functions for performing arithmetic.

Extended numeric data types also exist, such as LONGREAL, LONG INTEGER, and BCD REAL, and often their use cannot be avoided, as when greater precision arithmetic is needed for commercial arithmetic.

Rules:

- Avoid use of the BYTE data type. Instead, use either CHAR or a subrange of

0..255.

- Avoid use of WORD or ADDRESS data types. These are not particularly useful as numeric data types and are intended primarily for direct manipulation of memory, a technique that will be expressly forbidden!

- When using INTEGERs, create a 16-bit integer subrange in Pascal/3000 and use this to maintain compatibility with microcomputer Pascals.

- When using REALS, use the 32-bit data type in Pascal/3000. An exception to this might be when using 80-bit reals in Apple /// Pascal, or if 8087 support (80-bit real) is available on an MS-DOS machine. However, using extended precision on microcomputers usually means doing all arithmetic with procedure calls rather than with normal assignment statements.

- Limit ENUMERATED TYPES to 255 elements. This is to conform to a limitation on Pascal/MT+.

- Limit the range of SUBRANGE types to -32768..32767.

- Limit SETs to 255 elements. If you are working mostly with Apple Pascals, then this may be increased to 512 elements, but MT+ only supports 255.

- Avoid placing SETs as a component of an array of a record when using Pascal/MT+, otherwise the data stack may be consumed by unused portions of the record. Only HP and Apple Pascals allow packing to the minimum number of bits, but even here the structure will be "padded" to the next word level;

- Restrict all usage of "conformant" array types to modules which are intended to duplicate the functions of one Pascal in another dialect. An exception to this is the STRING type.

- Limit the maximum size of STRINGs to 255 characters.

- When commercial arithmetic requires more precision than INTEGER allows, use:

  LONGREAL in Pascal/3000,

  LONG INTEGER in Apple /// Pascal,

  REAL BCD numbers in Pascal/MT+.

  In IBM PC Pascal there is no support for signed numbers greater than 6 digits. However, an external module giving 8087 support may be available.

- Isolate all routines using high precision numeric types and be prepared to totally rewrite for each dialect.

Packing

The original definition of Pascal included the concept of "packing" data into a format that required the smallest amount of actual machine storage that could represent an item of data. An example can probably best illustrate how this might be used.

For example:

Packing to the bit level:

```
var FOPTIONS : packed record
   DOMAIN               : 0..3;     {14:2}
   ASCII_BINARY         : 0..1;     {13:1}
   DEFAULT_DESIGNATOR   : 0..7;     {10:3}
   RECORD_FORMAT        : 0..3;     { 8:2}
   CCTL                 : 0..1;     { 7:1}
   TAPE_LABEL           : 0..1;     { 6:1}
   NO_FILE_EQUATE       : 0..1;     { 5:1}
   reserved             : 0..63;    { 0:5}
   end;                             {16 bits total}
```

requires only 16 bits of storage! While a construct like the example given is mostly useful for interfacing to a native operating system, no doubt other uses for this will also seem desirable. Unfortunately this is one area of greatest weakness in most microcomputer Pascals. You may declare the above in HP or Apple Pascals and produce the desired effect, but not in IBM PC or MT+ Pascals. However, IBM PC and MT+ partially make up for this by supporting bit level manipulation either directly, in the case of MT+ with its bit-oriented intrinsics, or indirectly through SETS (i.e., var BITS : set of 0..15;). In addition, MT+ and IBM PC Pascals will automatically allocate storage for CHAR arrays and integer subranges to the minimum number of bytes required, rather than to the nearest word.

All Pascals allow you to *declare* PACKED variables. However only HP and Apple Pascals actually perform packing.

Rules:

- Avoid defined records designed to pack components into a single word or byte.
- If you must use such structures, try to isolate their usage into a few easily modifiable procedures.
- Do declared structures and arrays as PACKED; even when one version of Pascal may treat the declaration as a comment, other versions, which support packing, may require the declaration to perform properly.

Procedures and Functions

Most Pascals place little restriction on the size of a procedure or function, but Apple Pascal limits the maximum size of a procedure to about 1200 16-bit words of p-code. In practice this may be several pages of source code, so this is not as great a restriction as it may seem. There are also limitations on the number of procedures and functions that may be declared in a single compiled module; this number may vary from 127 to 255 depending upon the release number of the Pascal.

One of the more interesting features of Pascal/3000 is the ability to return a structured data type, like an array or a record, from a function. Unfortunately, none of the microcomputer Pascals examined here support this, and this feature should never

be used!

Rules:

- Keep PROCEDUREs and FUNCTIONs reasonably small; a limit of two to three pages of source code should keep you within most limits.
- Keep the number of PROCEDUREs and FUNCTIONs within each separate compilation unit to less than 128.
- When using Pascal/3000, do not declare functions which return structured data types.

Statements

It is possible, in all of the Pascal dialects discussed here, to obtain the actual memory location of declared variables. In some Pascals it is also possible to perform arithmetic directly on pointer types. Using such techniques should be highly restricted, at best, if not totally forbidden.

Various extensions are available in some Pascals for use with boolean expressions. For example, in many Pascals all parts of a boolean expression are evaluated, even after the "outcome" has been presumably determined.

For example:

```
function INCR (A, B: integer) : integer;
   begin
   INCR := A + B;
   end;
...
...
if (AMOUNT > 1000) and (INCR(AMOUNT,100) > 5000)
```

will *always* perform the call to "INCR". Some programs, use this "side effect" to advantage, but extended boolean operators like "&" for "AND" and "|" for "OR" may terminate the evaluation of an expression before the side effect is invoked: this effect is similar to invoking the "PARTIAL_EVAL" compiler option in Pascal/3000. These operators should be avoided.

NOTE: When using IBM PC Pascal *no assumptions should be made as to whether all parts of a boolean expression are evaluated* sometimes all parts of an expression may be evaluated or sometimes the optimization process may cause evaluation of an operand to be skipped. For this reason, if IBM PC Pascal is to be one of the targetted versions of an application, you should definitely avoid creating expressions designed to use a side effect.

Some Pascals (such as IBM PC Pascal) support the CYCLE and BREAK statements, which were "imported" from the C language, and provide extended control of FOR and WHILE loops. These should be avoided. In Pascals which do not have these statements, CYCLE and BREAK may be converted as follows:

Example:

Simulating BREAK and CYCLE with GOTO:

```
FOR I := 1 TO N DO
   BEGIN
   IF MONTH[I].AMOUNT = 0
   THEN {CYCLE} GOTO 1 {the next iteration of the loop}
   ELSE
      BEGIN
      TOTAL := TOTAL + MONTH[I].AMOUNT;
      IF TOTAL >= LIMIT
      THEN {BREAK} GOTO 2; {the next statement}
      END;
1: END;                    {THE NEXT ITERATION OF THE LOOP}
2: ...                     {THE NEXT STATEMENT}
```

In the FOR statement, limitations should be placed on the control-variable, even if a particular Pascal dialect does not require it. The control-variable:

- should be an ordinal type.
- should not be a component of a structure.
- should be locally declared at the same level as the FOR statement which uses it.
- should not be a reference parameter to a procedure or function.

Note that while Apple /// and IBM PC Pascals are relaxed about this, Pascal/3000 and Pascal/MT+ are not.

In the CASE statement, limitations should be placed on the case selector variable, and the case constants:

- the case selector should be an ordinal type (CHAR is also acceptable).
- the case selector should not be a component of a packed structure (an element from a PACKED ARRAY OF CHAR is usually safe).
- case constants should not be specified as a subrange, i.e., "'A'..'Z'".
- the range expressed by the smallest to the largest case constant should not be excessive, as many Pascals build a "jump table" of case constants as part of the object code.

Rules:

- Do not use pointer arithmetic and tricky pointer types designed to allow direct manipulation of memory.
- Avoid extensions to boolean operators.

- Avoid reliance upon "side-effects" from the evaluation of boolean expressions.
- Avoid statements "imported" from another language, like BREAK and CYCLE.
- When using the FOR statement, limit the scope of the control variable.
- When using the CASE statement, limit the complexity of the selector variable and limit the range implied by the case constants.
- When using the OTHERWISE part of a CASE statement, some Pascals may use "OTHERWISE" or "ELSE", but a few Pascals, such as Apple // Pascal have no equivalent.

4. A Few Areas of Specific Concern

In practice, I have found the following areas to be the most difficult when programming for portability. This is because, unlike the general implementation guidelines which can be governed by limiting or omitting specific features, the following areas usually can not be simply omitted and specific changes must be made when moving from one Pascal to another. In some cases, the best solution will be to develop custom library routines for each dialect.

String Handling

Strings are, in effect, variable length packed arrays of CHAR. This data type is most useful for handling terminal inputs and outputs, but is virtually required when dealing with TEXT files, where the length of an input or output line will vary in length. In practice, you will find that extensions for STRINGs are the most compatible extensions among the various Pascals.

Because HP Pascal/3000 allows the return from a function to be a structured data type, like STRING, it is easier to prepare routines in Pascal/3000 to perform the same string functions as in microcomputer Pascals.

Converting numerical data to string (or character) format is highly non-standard, usually involving "writing" a variable into a string. No Pascal contains editing features similar to the PICTURE clause of COBOL. One of the first custom modules that should probably be written for each Pascal dialect is an "EDIT" package to handle this.

Rules:

- Attempt to conform to the basic UCSD string intrinsics. In Pascal/3000 it will be easiest to prepare a module that is syntactically the same as these intrinsics. Note that HP string functions are a superset of all the others.
- In IBM PC Pascal, be aware that there are *two* types of strings, a distinction that does not exist in the others.
- Develop a module with a standard interface to handle "editing" of numerical data into strings, and write a version for each Pascal dialect.

Micro-equivalents to HP Pascal/3000 string intrinsics:

```
setstrlen  - no microcomputer equivalent
str        - will become COPY
strappend  - similar to CONCAT
strdelete  - will become DELETE
strinsert  - will become INSERT
strlen     - will become LENGTH
strltrim   - no microcomputer equivalent
strmax     - similar to SIZEOF
strmove    - similar to COPY used within INSERT
strpos     - will become POS
strread    - no microcomputer equivalent
strrpt     - no exact equivalent; FILLCHAR might be used
strrtrim   - no microcomputer equivalent
strwrite   - no exact equivalent; STR might be used
```

File Access

Pascal/3000 provides for the most varied (and sometimes most useful) extensions to file access over the orignal Wirth definition. In most cases there are equivalent ways of doing the same things in microcomputer Pascals, but occasionally there will be a feature that will be difficult to duplicate, or should be avoided altogether.

Opening files:

All of the Pascals allow actual file designators (i.e., file names) to be associated with a formal Pascal file designator, usually as part of an "open" procedure; i.e. RESET, REWRITE, and (sometimes) OPEN. Normally the "open" procedure which is used on a file will indicate the type of access desired for that file; i.e., read, write, or read-write (which is usually equivalent to "direct" or random access). In Pascal/3000 this is definitely the case, but in microcomputer Pascals, this is not exactly an accurate way of looking at things; rather, the type of open procedure really determines whether a file already exists (RESET) or if the file is new (REWRITE). This last point will be elaborated in the section under "Temporary files".

Examples:

Opening a file for direct access:

```
open (MYFILE, "data.ptest.dev");    {Pascal/3000}
reset (MYFILE, "dev/ptest/data");   {Apple /// Pascal}
assign (MYFILE, "B:data");          {Pascal/MT+}
reset (MYFILE);
```

Direct access:

The direct access to files is an extension that is nearly the same in all of the Pascals examined. Access is always by record number, and record numbers always start with zero. The only real concern here is that Pascal/3000 has extended the READ and WRITE procedure to permit them to be used with any type of file, including direct

access. Normally READ and WRITE can only be used on TEXT files, and GET and PUT must be used on all other file types. When using Pascal/3000 you should avoid the extended usage of READ and WRITE.

Examples:

Locating and reading a direct access file:

```
seek (MYFILE, RECNO);                    {Pascal/3000}
get (MYFILE);
seek (MYFILE, RECNO);                    {Apple /// Pascal}
get (MYFILE);
seekread (MYFILE, RECNO);                {Pascal/MT+}
```

Appending to a file:

One would imagine that appending to an exsiting file would be a basic operation included in all Pascals. This turns out not to be the case. Only Pascal/3000 supports an append access mode (by opening a file with APPEND). In other Pascals it may be necessary to open the file (with RESET) and read through the file until EOF is reached, or even to open *two* files (one with RESET and one with REWRITE) and copy from one into the other until EOF; then you may start appending to the second file. Historically, this situation came about because the microcomputer operating systems allocated disk space on a contiguous basis, and an append operation could conceivably overwrite the physical file following the one you are appending to.

Sensing end of file:

Normally sensing end of file is not a problem, but in Pascal/MT+ if a "typed" non-TEXT file is being read then you cannot rely on the standard EOF function. This is because end of file information in MT+ is based on the number of sectors used by a file, not the exact number of bytes. In Pascal/MT+ it may be necessary to use a special end-of-file record.

Closing files:

Closing a file seems straightforward enough, but there are a few considerations to be taken into account. There is the issue of saving a file, or deleting it. Also, most Pascals permit you to "reopen" a file with RESET, which closes the file and then reopens it repositioned to the beginning. Various close options are most important when dealing with temporary files, a topic which will be covered next.

Examples:

Closing a file and deleting it:

```
close (MYFILE, purge);                   {Pascal/3000}
close (MYFILE, purge);                   {Apple /// Pascal}
close (MYFILE, IORESULT);                {Pascal/MT+}
purge (MYFILE);
```

Temporary files:

All of the Pascals support some form of a "job temporary" file. Normally temporary files are created by opening a file which does not exist with REWRITE, but there are minor variations to this in each Pascal.

In Pascal/3000 a temporary file may be opened by omitting the filename in the REWRITE statement, but in this case the file can not be saved. If a filename is given, then either the file should not exist, or there should be a file equate issued before executing the program making the new file TEMP. Here the temporary file may be saved by using the SAVE option with the close statement.

In Apple /// Pascal, a file opened with REWRITE is *always* temporary, preserving the contents of any existing file. Here, the file must be closed with the LOCK option to make it permanent and delete the old file.

In Pascal/MT+, temporary files are opened with REWRITE with the file name ommitted. The file is assigned a special "system" name with a numeric suffix. Closing the file will not delete it, so presumably it may be renamed after the program terminates.

Rules:

- Be prepared to customize all opening and closing of files.
- Be prepared to customize direct access of files, but don't be afraid to fully utilize this.
- When using Pascal/3000, avoid using the extended versions of READ and WRITE which permit these to be used on non-TEXT type files. Instead use GET and PUT.
- Avoid extensive use of IORESULT. If you find life easier by checking the I/O status, define a series of constants, which can be changed to suit each particular operating system, and compare IORESULT against these.
- When using "typed" non-TEXT files in Pascal/MT+, do not rely on the EOF function; instead use a special record to mark the end of the file.
- If you wish to append to an existing file, you will have to develop a custom module for each of the microcomputer dialects. In some cases this may require the reading of a file until EOF becomes true.
- If you must use low level access of disk files, develop a module with a common interface, but be prepared to write a specific implementation for each computer. A possible exception to this might be BLOCK I/O in MT+ and Apple Pascal.
- If you require keyed access, the only Pascals that attempt this are Pascal/3000 and Apple /// Pascal and then only with external intrinsic support. No doubt similar packages might be found for PC and MT+.

Micro-equivalents to HP Pascal/3000 I/O intrinsics:

```
append      - no equivalent
close       - same, with some variations
eof         - same; may vary with non-TEXT file types
eoln        - same
fnum        - no equivalent
get         - same
linepos     - no equivalent
maxpos      - no equivalent
open        - will become RESET
overprint   - no equivalent
page        - same
position    - no equivalent
prompt      - become a simple WRITE;
put         - same
read        - used for TEXT files only
readdir     - no exact equivalent
readln      - used for TEXT files only
reset       - same, with some variations
rewrite     - same, with some variations
seek        - same; may become SEEKREAD or SEEKWRITE
write       - used for TEXT files only
writedir    - no exact equivalent
writeln     - used for TEXT files only
```

Screen Handling

Normal Pascal I/O was not designed for use with interactive terminals. All Pascals here have circumvented certain problems by defining special file types for interactive type devices. Normally this is transparent to the programmer if the standard files INPUT and OUTPUT are used. However, no provision has generally been made for anything more sophisticated than reading and writing variables in serial fashion. For most applications, this "character mode" approach will work quite well.

However, if you wish to design "screen mode" interaction you will have to resort to programming a custom module for each associated "terminal type" for each Pascal. This module should use the same procedure calls in each version, and may include functions like:

```
Home_Cursor          - home cursor to screen upper left
Clear_EOS            - blank to end of screen
Clear_EOL            - blank to end of current line
Goto_XY (Col, Row)   - move cursor to Row, Col
Insert_Char          - insert a blank at the cursor
Insert_Line          - insert a blank line at the cursor
Delete_Char          - delete the character at the cursor
Delete_Line          - delete the line the cursor is on
and so forth...
```

A good model for such a package may be found in the UCSD Screen_Ops unit, however most users will have no trouble designing a reasonable list for themselves. There are really only two considerations. First, whatever you devise, use it consistently, otherwise it will loose its value as an internal standard. Second, be wary of using techniques that interact at the character (or keystroke) level. While character based interaction will work effectively on all microcomputers, it will not work particularly well on the HP 3000. This is because the HP 3000 is designed to normally pass input from a terminal read upon the receipt of a carriage return, and while this may be over ridden by an FCONTROL option, character by character on even a lightly loaded system is normally slower than most people can type.

Rules:

- Whenever possible, use normal Pascal I/O for screen interaction. While this will limit you to line and prompt oriented screens, this will always be the most portable solution.

- Avoid character based interaction. While this may work well on microcomputers, MPE does not support this without frequent and annoying delays between the typing of a character and its appearance to the requesting program.

- Keep the screen oriented routines isolated so they can be customized for different terminal or console types. On HP computers (the 3000, 120, and 150) one can probably assume that an HP terminal is being used. On the Apple /// use the .CONSOLE driver. On the IBM PC running PC-DOS 2.0 or higher, use the ANSI.SYS driver.

Heap Management

Sometimes it is advantageous to allocate work space dynamically while a program is executing. This may be used for handling a previously unknown number of variables, or used perhaps because this feature allows for the elegant handling of "linked lists" and other complicated structures. All Pascals support this: each new occurrance of a variable is created using the NEW procedure, which allocates space for it on the "HEAP", a specially reserved area of the data stack. Items added to the HEAP can only be accessed through pointers, and normally the programmer has no knowledge of exactly where in memory a variable might reside.

Many programs can function if simply given the ability to create items when needed. Problems relating to portability occur if a program also must periodically remove items from memory to free space. Wirth Pascal defined DISPOSE as the procedure which removes an item from the HEAP. However, this presents a problem when space must be compacted from the holes left in memeory by disposed items. Two strategies have emerged for handling this. Some Pascals perform "garbage collection" on the heap when NEW requires it. Other Pascals, such as Apple Pascals, require the programmer to perform this chore, by MARKing a position in the HEAP, then using RELEASE to shrink the stack to a prior state. The two strategies are generally incompatible. Fortunately Pascal/3000 supports them both. Pascal/MT+ also will support both methods, but IBM PC Pascal only supports NEW and DISPOSE.

Rules:

- NEW is standardly implemented. However, not all Pascals support tagged variants.
- Getting things off the heap is not standard. While DISPOSE is in HP, PC, and MT+ Pascals, Apple /// does not support this feature. While MARK and RELEASE is in HP and Apple Pascals and can be used in MT+ with a short user routine, IBM PC Pascal cannot support this feature.

5. Constructing Common Intrinsics

All of the Pascals discussed here allow program segements to be compiled separately from a main program, and included using a separate "linkage" program. In fact, virtually all microcomputer Pascals have implemented various "standard" features of Pascal as library modules, which must be specifically included within a program if the features are to be used.

However, Pascal/3000 and Apple /// Pascal also allow the separate compilation of "intrinsic" procedures, which may be placed in library code files and bound to a program when the program is executed.

While a detail discussion of how this is done in each of the Pascal dialects is beyond the scope of this paper, a few examples using Apple /// Pascal and SOS intrinsics can illustrate the general concept.

Many MPE intrinsics have similar SOS counterparts; such as:

```
FOPEN      SOS_Open
FGETINFO   SOS_Get_Info
FREAD      SOS_Read
FREADDIR   SOS_Set_Mark followed by SOS_Read
FPOINT     SOS_Set_Mark
FWRITE     SOS_Write
FWRITEDIR  SOS_Set_Mark followed by SOS_Write
FCLOSE     SOS_Close
GETDSEG    SOS_Request_SEG
FREEDSEG   SOS_Rel_Seg
```

While the parameter lists vary extensively between MPE and SOS intrinsics, it is entirely feasible to construct a "HPFILES" UNIT in Apple /// Pascals which looks and performs similar to MPE intrinsics. One may wish do this because using SOS intrinsics directly in Apple /// Pascal is faster and uses less stack space than using the built-in Pascal I/O functions, and it also is easier to duplicate Pascal/3000 features like APPEND access to a file if one resorts to SOS file intrinsics. No doubt a close familiarity with similar low-level support in Pascal/MT+ and IBM PC Pascal could produce similar results.

6. Microcomputer Environmental Similarities to MPE

Some microcomputers support environments uncannily close to a "single tasking

MPE" environment; i.e. 64Kb separate code and data stacks; a hierarchical file system which can simulate "groups" and "accounts"; and program and system code libraries.

Program data and code stack sizes in Apple /// Pascal and IBM PC Pascal meet or exceed those available under MPE. For Apple // and CP/M-80 systems, expect less than half the amount available under MPE. However, both of these Pascals have operating systems comming in common useage which support at least 128Kb (Pro-DOS and CP/M+), so expect about three quarters of the space.

The file systems with greatest compatibility with MPE are MS-DOS (version 2.0 and higher) and Apple /// SOS (and soon, Pro-DOS on the Apple //e). Unlike MPE, which supports only two levels of file directories, MS-DOS and SOS support nearly unlimited directory levels. (There is a practical limit of the length of a file pathname which may be specified when a file is opened.)

The ability to combine separately compiled routines is available on the HP and in all of the microcomputer Pascals covered here. The approach used is virtually the same; a program like the segmenter is available to combine modules into a new runnable code file.

The library facilities closest to MPE are found only in Apple Pascals, and in the UCSD p-System. Unlike MPE, which supports three levels of libraries, system, account, and group, Apple Pascals support only two levels, system and program. However the program library may be a list of library file names, which means that an Apple program library can effectively be up to five separate library files, which may reside under any accessable directory structure.

7. Conclusion

Depending upon the complexity of an application, the same kernel program can be implemented with little change on an HP 3000 and a microcomputer. However, if the same program must run on several different microcomputers then some modification for each microcomputer will be required (perhaps as much as one or two months).

The key point one should have formed from this paper is that one must adopt a long range strategy so that once a basic set of restrictions, techiniques, and modules are specified and developed for each microcomputer, they may be reused in future applications.

As a final comment, here is my ratings of the degree of compatibility between the Pascal dialects covered here, from highest to lowest:

- HP Pascal/3000 to Apple Pascal
- HP Pascal/3000 to Pascal/MT+
- HP Pascal/3000 to IBM PC Pascal
- Pascal/MT+ to IBM PC Pascal
- Apple Pascal to Pascal/MT+
- Apple Pascal to IBM PC Pascal

BIBLIOGRAPHY

"Pascal User Manual and Report", second edition,
Kathleen Jensen and Niklaus Wirth,
●1974 by Springer-Verlag

"Pascal/3000 reference manual",
●1981 by Hewlitt-Packard Company

"MPE Intrinsics reference manual",
●1981 by Hewlitt-Packard Company

"KSAM/3000 reference manual",
●1981 by Hewlitt-Packard Company

"IBM PC Computer Language Series Pascal Compiler"†,
●1981 by IBM,

"Pascal/MT+, language reference manual"†, release 5
●1981 by Digital Research

"SpeedProgramming Package User's Guide"†, release 5.2
●1982 by Digital Research

"Apple /// Pascal Programmer's Manual", volumes 1 & 2,
●1981 by Apple Computer

"Apple /// Pascal Technical Reference Manual",
●1983 by Apple Computer

"SOS Reference Manual", volumes 1 & 2,
●1982 by Apple Computer

"RPS Programmer's Manual"†,
●1983 by Apple Computer

"The Pascal Handbook",
Jacque Tiberghien,
●1981 by SYBEX Inc.

† Manual not available separately; requires the purchase of a software product.

1 9 8 4
HP3000IUG
ANAHEIM
CONFERENCE

AUTHOR BIOGRAPHY

Author: Kenneth C. Butler
Manager, Technology
Imacs Systems Corporation

Title: Portable Development Between the HP 3000 and a Microcomputer Using Pascal.

Biography: Kenneth C. Butler is the Manager of Technology for the Imacs Systems Corporation, where he is leading a research and developement program involving the use of microcomputers. He received his bachelor's degree in psychology from Michigan State University in 1968. After a tour of duty in the Air Force, he has worked in the field of data processing since 1972, starting as an IBM Assembly language programmer in a downtown Los Angeles service bureau. His first exposure to an on-line system was in 1977, when he became the system manager of a Tandem Non-Stop system. Between 1980 and 1983, he was employeed by the Twentieth Century-Fox Film Corporation, where he was the system manager for four HP 3000's and acted as the primary technical applications specialist for the TCF Branch On-Line System, one of the first large systems to use the Transact programming language. He became actively involved in office automation and the corporate use of microcomputers while at Twentieth Century-Fox, and owns his own personal computer.

COMPARATIVE MERITS OF A CENTRAL DATA BASE AND REPLICATED DATA BASES FOR AN INTEGRATED INVENTORY CONTROL SYSTEM

CLEARING THE FOREST

The guiding principle governing distributed information systems is optimum functional distribution. With functional distribution, only selected processing functions take place within a single element of a total system, while other functions are distributed to other system elements. The goal is to locate specific elements physically close to the users of the functions being distributed. In significant contrast, the concept of centralized functionality dictates that all computing and data storage resources are physically located in one processing element. Centralized functionality presupposes that users must come to the computer or, at best, will be remotely attached to the almighty source.

The last decade has rendered a variety of related buzz words with overlapping and confused meanings. These include terms like "distributed processing," "distributed systems," "distributed networks," and "distributed data bases." The sad part of all of this is that these terms now have no universally understood definitions. They are used by authors and industry professionals differently. There has, however, been a fortunate recent trend to standardize some of these terms,[1] and in this paper I will attempt to use terminology which reflects this move toward standardization.

A distributed information system can be defined as "a coordinated set of information processing capabilities implemented in two or more relatively independent resource centers such as computer sites, terminal locations, and so on."[2] In my view, there are three substructures which combine to form the total structure of a truly distributed information system. These are:

1. A distributed processing substructure that reflects "a technique for implementing one logically related set of information processing (application-related) functions within multiple physical devices."[3]

2. A distributed data base substructure which implies "a single logical data base which has been implemented in more than one physical segment, attached to more than one information processor."[4]

3. A networking substructure consisting of "the hardware and software functions which support the definition, establishment, and use of facilities for data movement among (usually physically separated) information system components."[5]

Theoretical consideration of one or any combination of these substructures would result in a work of great magnitude. A practical consideration (i.e., related to a user application within a specific hardware/software environment) could be similarly monumental. Therefore, my intention is to present a practical aid to help designers of automated inventory control systems make key decisions related to data base distribution. The impetus of this paper will focus on substructure 2 above, applicable whenever a solution involves a distributed data base. All cases will deal only with solutions which have been (or could be) implemented on HP-3000s using IMAGE and DSN/DS3000. All of the processors configured in this paper are of the same type (HP-3000 series), with all of the CPUs comprising any one configuration running under the same MPE operating system version, etc. In these examples, then, we will encounter no compatibility problems related to distribution of the work load. Many of the cases considered here are now operating using the solutions described. No companies will be referred to by name, and some configurations have been altered slightly (in order to heighten the effect of the solution).
Specifically, then, I will present cases involving wholesale distributors operating from multiple locations using HP-3000s to specifically control all of the automated Inventory applications, which include Order Entry, Stock Allocation, Order Filling/Shipping, and Warehouse Transfers, as well as other general considerations specifically related to supply and demand in the wholesale distribution business. The solutions I will examine are:

I. Distributed Data Bases

   1. Partitioned Data Bases

      a. Geographically Partitioned

      b. Hierarchically Partitioned

   2. Replicated Data Bases

      a. Hierarchically Replicated

      b. Fully Replicated

      c. Horizontally Replicated

   3. Combined Partitioned and Replicated Data Bases

II. Centralized (Nonreplicated) Data Bases

I. Distributed Data Bases

The two fundamental structures for data base distribution are partitioned and replicated. Complex applications may warrant the combination of both partitioned and replicated structures.

1. Partitioned Data Bases

All data base design involves the development of a conceptual data base, which should parallel a logical schema of some sort. The conceptual data base will include all data of interest to an organization. Whenever a conceptual data base is separated into nonredundant sections (i.e., partitions) and spread across multiple information processors, a partitioned data base is formed. Once all of these partitions are attached to a single information processor, together they form a single logical data base which defines the sum of the partitions. The generic partitioning of a conceptual data base often results in the configuration depicted in Figure 1.

Despite the physical representation of this configuration, there is, at this point, no determination of where the partitioned data bases and replicated hardware should reside geographically. For example, data volumes and performance considerations alone could require such a partitioning, although all three partitions occupy the same location, with all hardware in the same computer room. The manner in which a data base is partitioned very closely correlates to the information processing structure of the distributed system. In one case, partitioning may be used to insure privacy and improve security by partitioning public information from sensitive data. In another, data may be partitioned in order to physically locate it close to where access requests for that data originate.

1.a. Partitioning Data Bases Geographically

One organization in the wholesale distribution industry operates two HP-3000/64s in a horizontally distributed system (i.e., where both CPUs cooperate as equals). Their particular configuration exemplifies the classic geographically partitioned data base. The company operates on a nation-wide basis within the continental United States. Demand for the items distributed is affected both seasonally and geographically. The conceptual data base for this company locates data elements logically, so that data base elements accessed heavily by East Coast users are partitioned with the Eastern Host, while the Western Host partitioning reflects the same mentality. This configuration is depicted in Figure 2.

This specific geographical partitioning is the most practical solution for the company because of its natural geographic grouping of data base access requirements. If elements accessed from intermediate locations had been equitably distributed between the two centers, this partitioning would have caused entirely too much traffic between centers to be practical. Perhaps the best rationale for this solution is the high cost of cross-country data transmission. However, even in those instances where cost of transmission is not dependent upon distance (e.g., packet network services), transmission volume remains a crucial factor. The system illustrated in Figure 2 reflects an application where the majority of accesses to each geographic data base partition will originate locally. However, provision was still made for access of

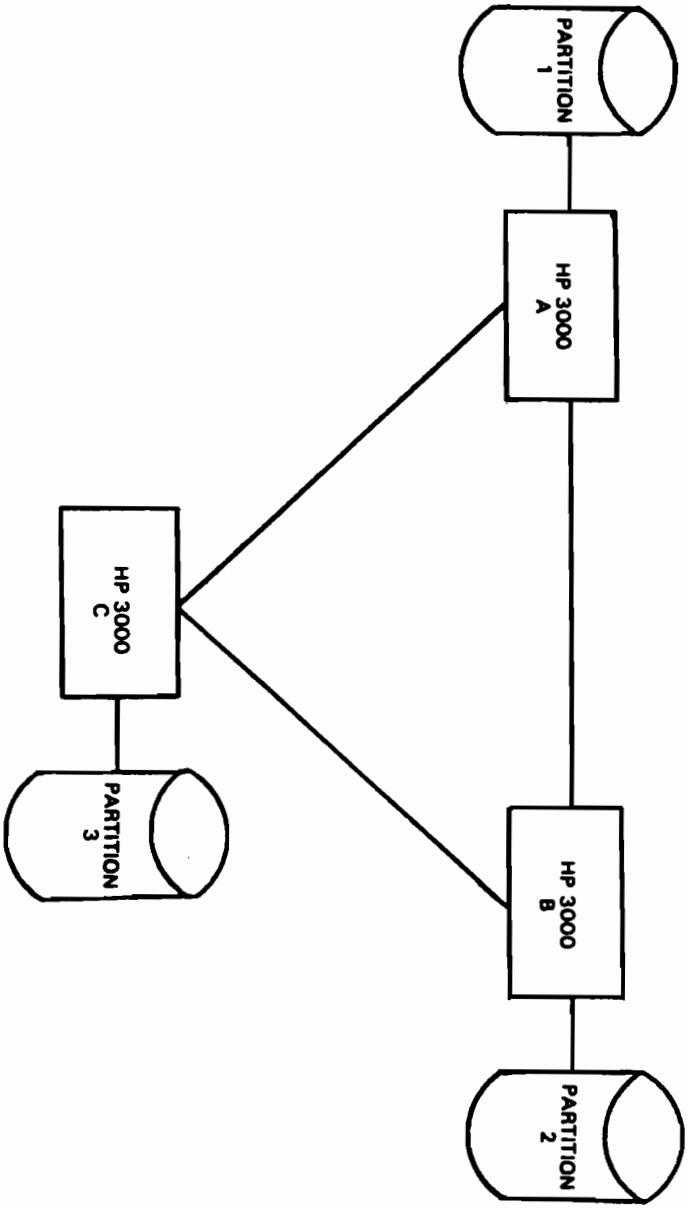


Figure 1: Generically Partitioned Conceptual Database

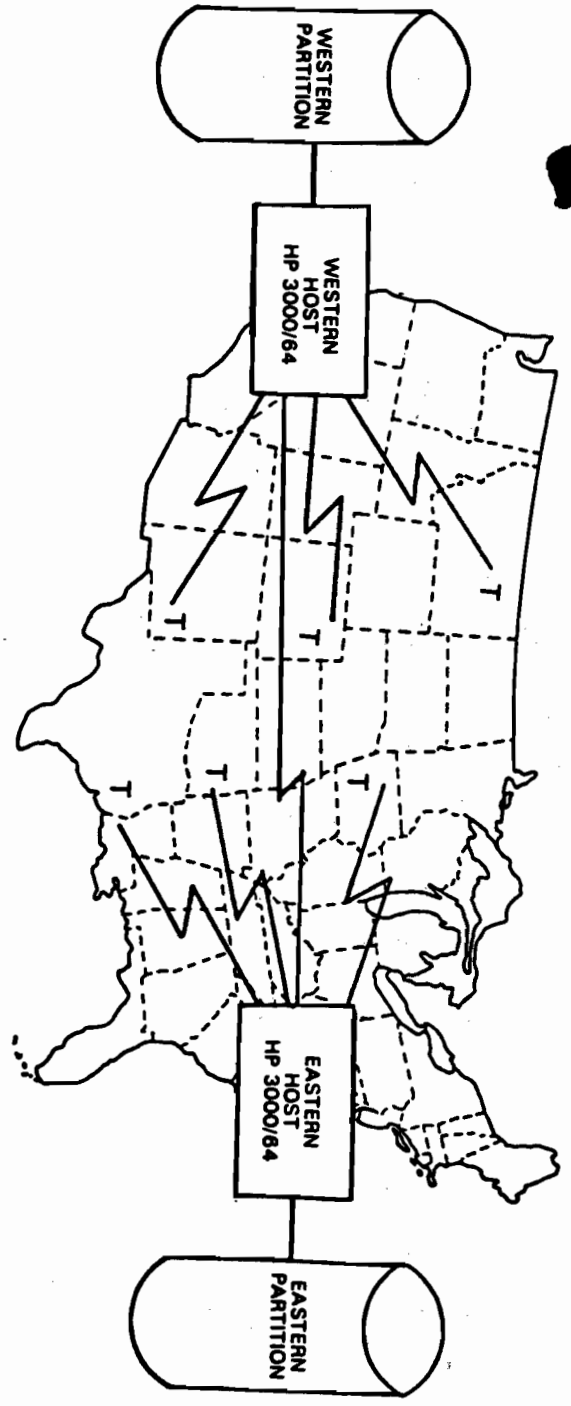


Figure 2: Geographically Partitioned Databases

the Eastern Host by the Western Division, and vice versa. For despite the excellence of the design of the partitioning, there are still a few inventory items which both locations stock commonly. Thus, there is some limited transferring of goods between divisions which had to be accommodated. Still, this design remains best for the way this company operates.

1.b. Hierarchically Partitioned Data Bases

Another partitioning methodology advocates the hierarchical distribution of data elements. Most examples of hierarchical distribution bring to mind applications in which the processing hierarchy corresponds to the organizational hierarchy (e.g., the highest-level host handles corporate-level processing and each subordinate division consolidation handles its own needs using other hosts or satellite CPUs). Organizations attempting to automate inventory control functions are discovering more and more instances ideally suited to the solutions afforded by hierarchically partitioned data bases. These instances are relatively new, coming with the advent of a breakthrough in modern automated inventory control systems called Distribution Resource Planning (DRP).[6] Traditional inventory control systems treat each distribution center in a network separately, so that each center must fend for itself and determine its own inventory requirements based upon its own isolated history of supply and demand. In the case of Regional Distribution Centers (RDCs) which supply satellite Distribution Centers (DCs), each DC traditionally places its own orders with an RDC. Items are often transferred between DCs and RDCs if inventory requirements necessitate such activity, which is expensive. With DRP, however, the requirements of all of the DCs supplied by any one RDC are considered cumulatively by that RDC, so that goods may be centrally dispersed in a more controlled, time-phased manner. Costly warehouse transfers are eliminated. In addition, DRP has proven that RDCs can get the appropriate goods to the appropriate DC within the required time frame. Consider the distribution hierarchy of Figure 3.

In this example, the individual inventory requirements for San Francisco and San Diego are summarized and added to the requirements for Los Angeles. Similarly, the individual requirements for Houston and New Orleans are summarized and considered to be part of the demand requirements for Dallas. At another level within this hierarchy, the individual (summarized) demand requirements for Los Angeles and Dallas are added together and planned for by the Chicago central supply facility, which must also plan for its own demand requirements. This arrangement is known as a "push" system, in which a common supply facility determines replenishment requirements for the distribution centers it supplies. This overall relationship lends itself very well to an automated solution involving hierarchically partitioned data bases, as seen in Figure 4.

In this configuration, the Chicago Host handles system-wide data, and Los Angeles handles data for itself as well as for San Francisco and San Diego. Dallas handles data for itself as well as for Houston and New Orleans. At the bottom (i.e., DC) level of the hierarchy, there will be no duplication of data. Each DC will have access to its appropriate RDC, yet no DC will have

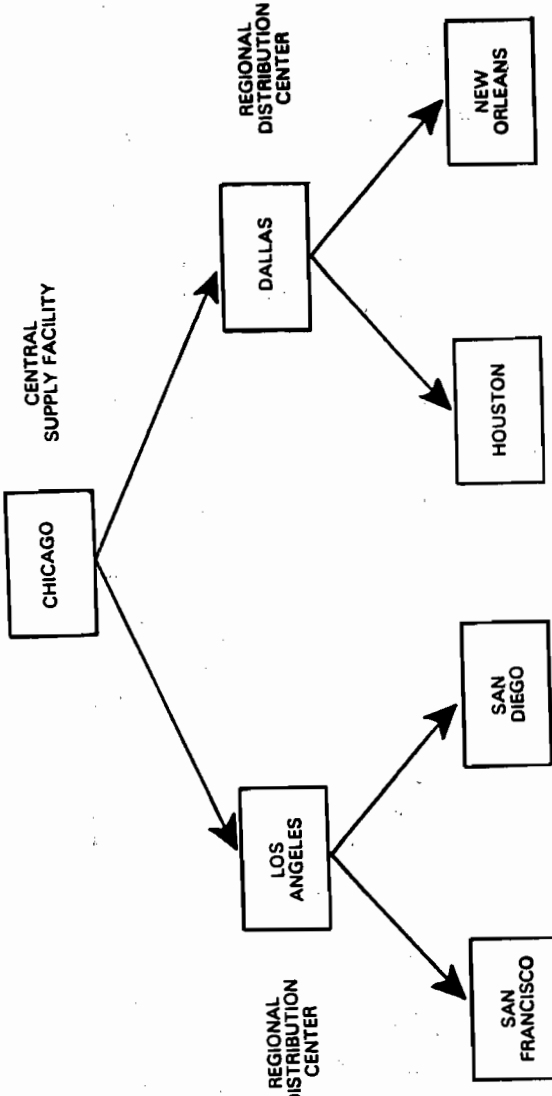


Figure 3: Hierarchically Organized Wholesale Distributor

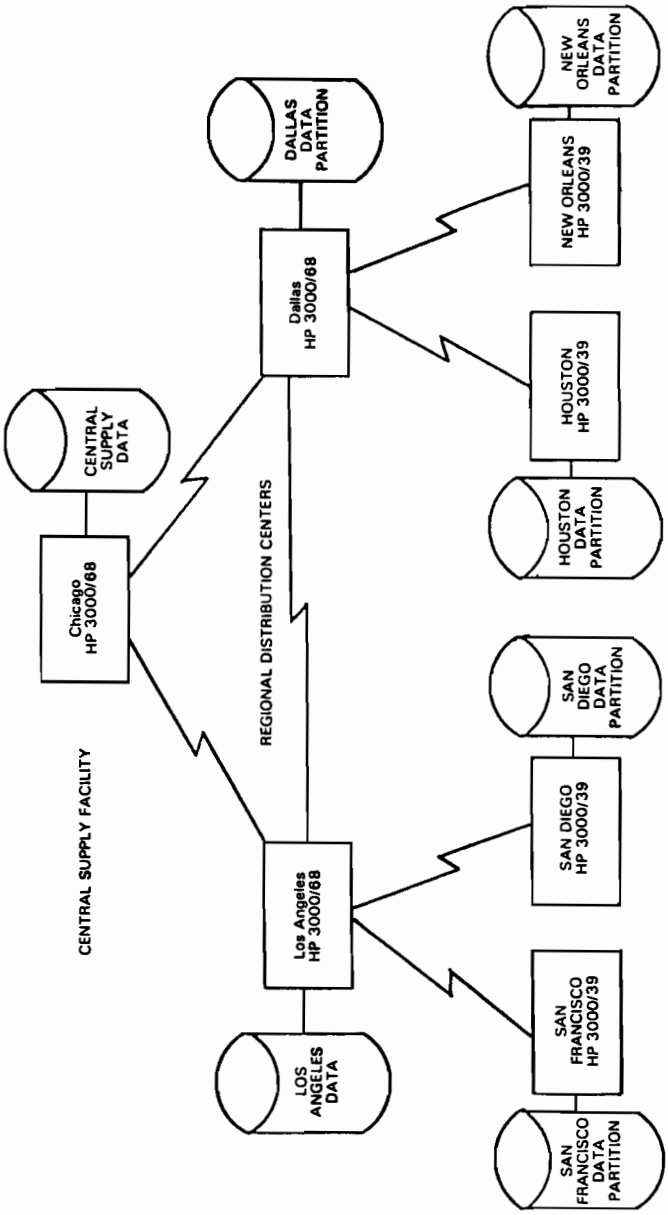


Figure 4: Hierarchically Partitioned Data Bases

data on site other than that pertaining to items distributed by it. Only at higher levels is there significant duplication of data. In effect, the data elements at all locations parallel the organizational hierarchy which has been successfully partitioned across the CPUs within the hierarchy. Because of this design, information can be exchanged both up and down the hierarchy. The central supply facility in Chicago must consolidate inventory demand requirements for all of the lower levels, while data must be distributed from each level to the next in a downward direction as delivery schedules are maintained. Because of this interrelationship of data traveling in both directions within the hierarchy, the overall system exhibits a structure which is truly partitioned and distributed hierarchically, rather than merely equaling the sum of separate independent data bases.

2. Replicated Data Bases

Another way to address distributed data bases is to place copies of all or parts of a whole company's data at multiple locations, resulting in a "replicated data base." Like partitioning, replication puts the data in the location where the access requests originate. The difference is that replication uses duplicate (i.e., copied) information, while partitioning enables the user to access original data elements. In instances where frequent and comprehensive data availability is required, replicated data bases provide a generally good solution. Essentially, there are two ways in which a data base can be replicated: hierarchically and horizontally.

2.a. Hierarchical Data Base Replication

A wholesale distributor operates with one central supply facility, two RDCs, four satellite DCs, and six nonstocking sales offices nation-wide. The central supply facility and both RDCs stock all of the items distributed by the company. Each satellite DC stocks approximately 50% of the items carried by the RDC which supplies it. In addition, certain sales offices may order only from specific DCs. The overall relationship is depicted by Figure 5.

Notice that New York supplies both Los Angeles and Pittsburgh, and all three carry identical items. Half of the items stocked by Pittsburgh are used to supply Cleveland, the rest are for Louisville. Half of the items Denver stocks are intended specifically for replenishment of Phoenix, with the remainder intended to supply Los Angeles. The Detroit sales office may order only from Cleveland, while the Tulsa sales office may order from either Cleveland or Louisville. The San Francisco sales office may order only from Los Angeles, but San Diego may order from Phoenix or Los Angeles, while the Las Vegas sales office must order only from Phoenix. This arrangement is known as a "pull" system, in which each distribution center determines its replenishment requirements and orders from a common supply facility. This odd, but real situation lends itself ideally to a solution involving hierarchically replicated data bases. This solution in depicted in Figure 6.

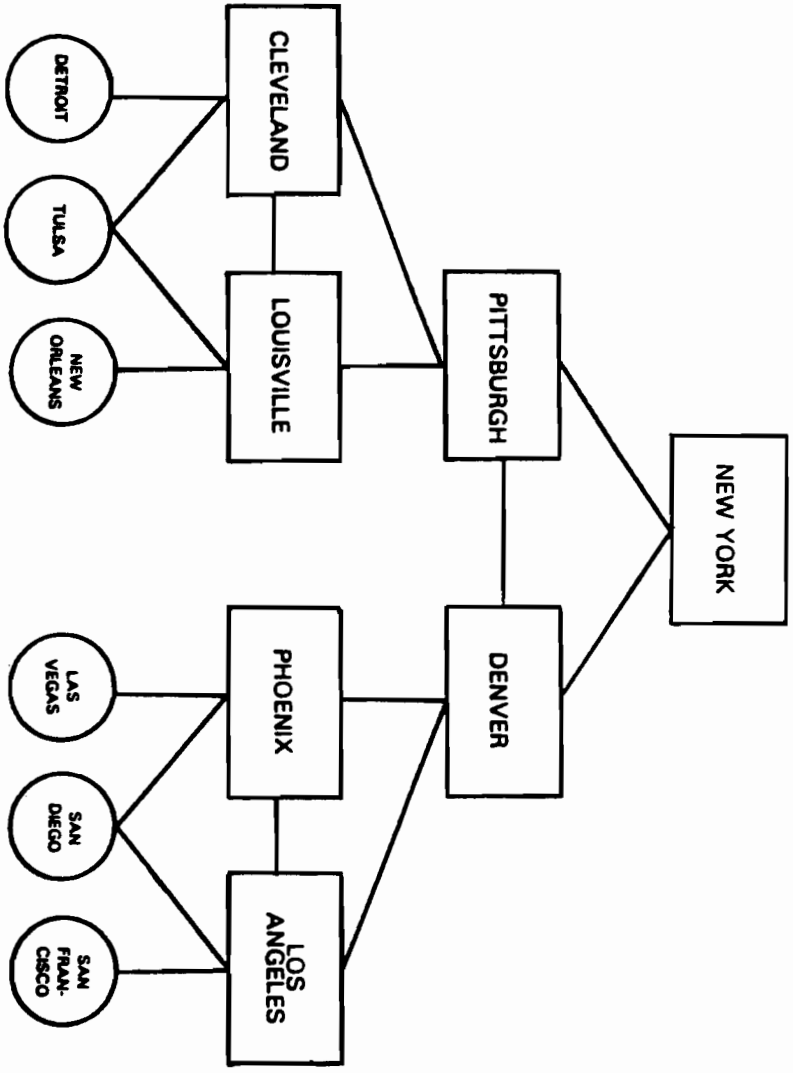


Figure 5: Hierarchically Organized Wholesale Distributor

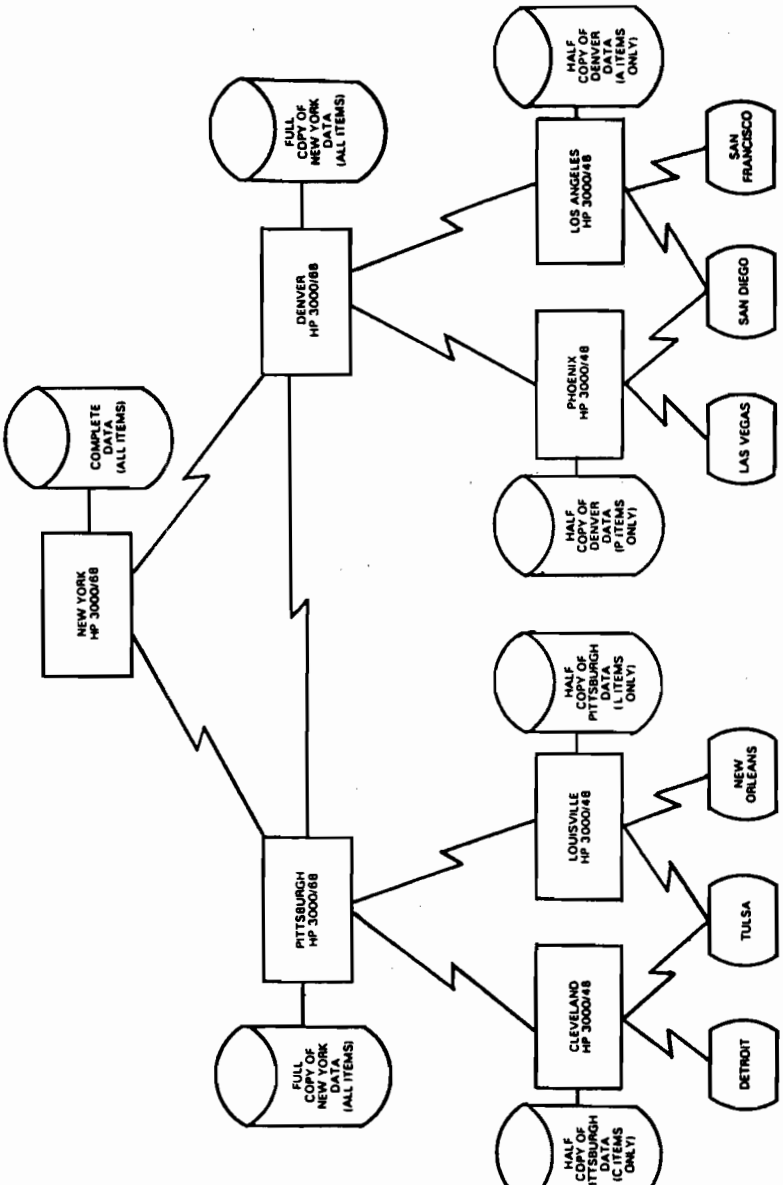


Figure 6: Hierarchically Replicated Database (Full and Partial Replication)

In this configuration, the New York host maintains a master data base containing the inventory control data for the entire organization. Pittsburgh and Denver also have complete physical copies of the New York data base. Cleveland has a partial replication of the master data base, consisting of the data related only to the items stocked by Cleveland. Louisville has a different part of the same master data base replicated, with only the items it ships. Phoenix and Los Angeles similarly have differing partial copies of the same master data base (i.e., the one shared by New York, Pittsburgh, and Denver). All of the replications at the lowest level are mutually exclusive, with no redundancy between them. Each local data copy, then, only replicates that data needed for its own operation. In contrast, the data bases for New York, Pittsburgh, and Denver are fully replicated. This turned out to be a big plus for this organization, because full replication served as back-up for the whole organization when there were problems.

2.b. Horizontally Replicated Data Bases

Another wholesale distributor of durable goods uses three HP-3000/68s as horizontally distributed hosts. Each distributed host maintains stock balances only for items available from its own physical location. In this sense, the data base is partitioned. However, in addition to this data, each host maintains product data consisting of product code, product description, and the location(s) stocking each product. Thus, the data for every host reflects a partial replication of the data of every other host. This configuration is depicted in Figure 7.

This arrangement allows orders for items not locally stocked to be validated and priced (for credit checking) before orders for them are sent to the appropriate stocking location. Items that do not exist anywhere within the inventory control system can be identified and rejected immediately. This instance of partial data base replication typifies a global directory or catalog of stocked items. A data base such as this, that is both horizontally replicated and partially partitioned, is generally rare. Most often data base replication is used exclusively in systems which are distributed hierarchically. Most horizontal system structures employ strict partitioning, with no overlapping data elements, or use separate, entirely independent data bases at each host location. Combination partitioned and replicated data bases are more common in other types of applications (e.g., airline reservation systems) than they are in inventory control applications.

II. The Centralized Data Base Approach

Generally, a centralized approach locates all computing and data storage in one processing element. Remember, though, that in one configuration, a centralized approach can involve multiple CPUs and even separated data bases in one location (e.g., the generic separation effected in Figure 1 could reflect three CPUs under one roof functioning together as one processing element). Also, the fact that all users of a system are served by the same data center supports the claim that linked CPUs and distributed data bases in one location add up to a centralized solution (although most industry professionals would call this a distributed solution).

Advantages

An approach involving one centralized data base is appropriate as long as its performance adequately satisfies the requirements of an inventory control system. In fact, any of the cases studied here could have used a centralized solution. Notably, there is no other alternative in instances where the conceptual (i.e., logical) data base cannot be partitioned. Most smaller distribution companies currently employ the centralized solution, even when shipping from multiple locations.

There are some advantages to a centralized approach. One is that you may avoid all of the disadvantages associated with the many alternative distributed approaches. Even in the centralized approach involving multiple CPUs and separate data bases in one location, you will eliminate many potential problems encountered when distributing data bases and processing resources across different locations.

Users who attempt to distribute data bases and processing resources across different locations may face a variety of problems. Many system designers tend to underestimate the difficulty of controlling access to multiple data bases. This problem is encountered whether or not linked CPUs occupy the same computer room. Although centralized data bases also require coordination and access control procedures, these can be handled with far less difficulty simply because everything happens on one computer. Security is similarly problematic, i.e., far less of a problem for one data base on one computer. System performance, back-up and recovery procedures, and data integrity maintenance across multiple copies which are partitioned and replicated in any number of ways are also requirements which can kill any efforts toward distribution. Associated costs most often end dreams of functional distribution. Initial costs required to replicate hardware, software, and personnel are only part of the picture. Facilities must be properly prepared (e.g., upgrades in electrical and air conditioning systems are just the tip of the iceberg). These issues have little to do with data base design. However, they may end the design of distributed data bases before it begins.

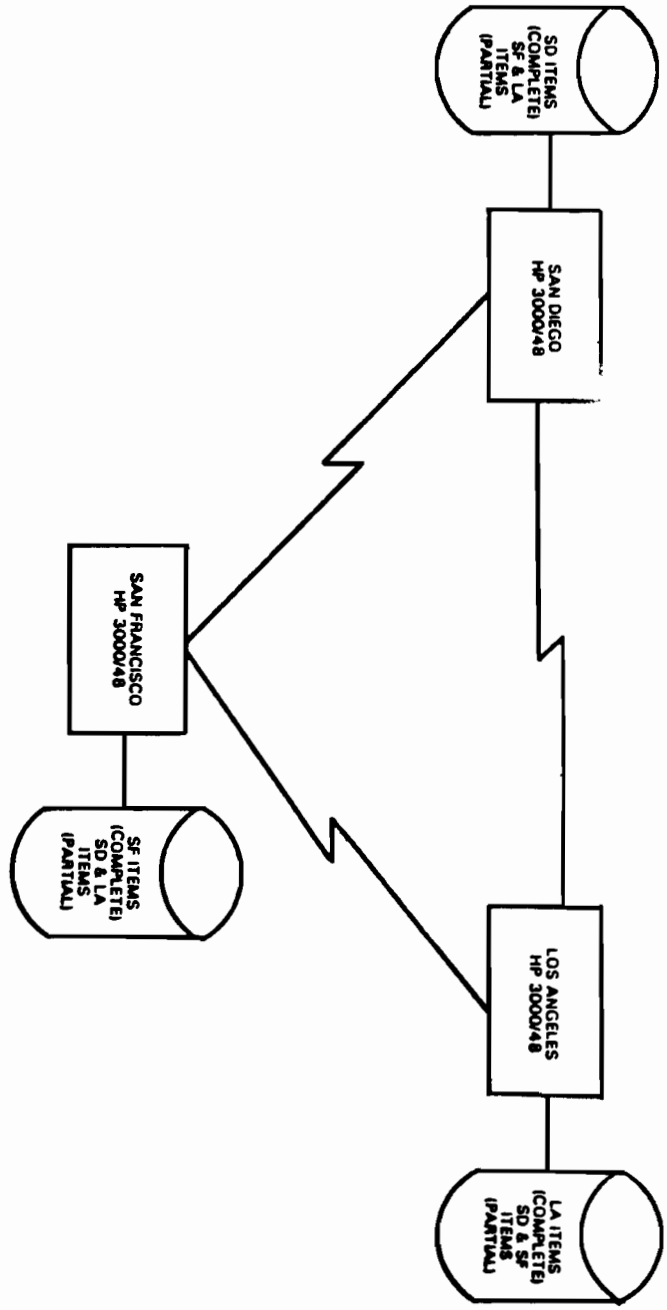


Figure 7: Combined Horizontally Replicated and Partitioned Databases

Disadvantages

On the other side of the fence, centralized data bases suffer from all of the disadvantages that are advantages for distributed data bases. Data base distribution will definitely improve data availability, as well as system survivability, given probable redundancy. Simply replicating data bases between CPUs will help to ensure that data is always accessible from some CPU, given the outstanding up-time reputation of the HP-3000. For organizations which must communicate among branches separated by great geographical distances, communication costs usually can be slashed. Finally, the modularity of distributed data bases means increased flexibility (e.g., certain users only need part of a system and may free-up hardware they would otherwise access by having their own data base and their own CPU).

Concluding Remarks

In its own effort to distribute certain of its organizational functions, Hewlett-Packard Co. has learned a valuable lesson:

> "The most significant lesson we have learned from our experience, however, is that there is no one best way to process data. Information systems must be designed to match the organization they support." [7]

To this, I add that in the next decade, the main breakthrough in information system design will come in the advances which will be made in the configuration of distributed information systems because of technological improvements in data base management systems, hardware, and telecommunications.

Acknowledgment

To Chuck MacKinnon, for persuading me to write this paper and (more importantly) for his masterful editing of it. Thanks, also, to Mark Buehler for executing the diagrams and Nancy Wild for word processing the text.

Notes

1. Although reflected in a number of sources, I feel that it is best exemplified by Grayce Booth in The Distributed System Environment: Some Practical Approaches (New York: McGraw-Hill, 1981).
2. Ibid., p. 255.
3. Ibid.
4. Ibid.
5. Ibid., p. 260.
6. For outstanding coverage of DRP, I strongly recommend Andre J. Martin's Distribution Resource Planning (New Jersey: Prentice-Hall, 1983).
7. Cort Van Rensselaer, "Centralize? Decentralize? Distribute?," Datamation, 25(4):90ff., Technical Publishing, New York, April 1979.

Bibliography

Booth, Grayce M., The Distributed System Environment: Some Practical Approaches (New York: McGraw-Hill, 1981).

Martin, Andre J., Distribution Resource Planning (New Jersey: Prentice-Hall, 1983).

Van Rensselaer, Cort, "Centralize? Decentralize? Distribute?," Datamation, 25(4):90ff., Technical Publishing, New York, April 1979.

JOHN HILL

John Hill is Manager of Distribution Systems for Computer Data Corporation, a nationwide company that develops and markets business software applications for the minicomputer industry. In his current position Hill is responsible for the development and enhancement of Inventory Control, Sales Orders, Purchasing, and Distribution Resource Planning (DRP) software. In addition he provides professional consultation in distributed systems configurations to CDc's clients. He holds an M.A. in Philosophy, as well as certificates in Business Computer Programming and Business Information Systems Design from California State University, Los Angeles. Hill has taught a variety of courses at LA-area community colleges. Prior to joining Computer Data Corporation, he worked for several manufacturing and distribution organizations, including Tandon Corporation, Sunweld Fitting Company, and Kinney Brothers of California. Past positions include Senior Manufacturing Systems Analyst, Database Administrator, Project Manager, and Data Processing Manager.

Hub of Systems Development and Documentation

Stephen M. Butler
Weyerhaeuser Company

INTRODUCTION

Throughout the traditional application development life cycle information about the project is generated phase by phase in a series of documents. Each phase has to adapt this information into its own format and documents. In some cases, this is a complete rework of the information and the way it is presented. The best examples are the transition from ANALYSIS to DESIGN and the documentation from DEVELOPMENT to PRODUCTION. Some claim that this last is neither valuable nor useful in its current form!

Dictionary support for the various phases of the application life cycle is not new. Generally, such support has been specific to each phase; but, the interfaces still are paper documents. Thus the dictionary support effort of earlier phases are duplicated for each subsequent step. Use of the dictionary for the interface will allow the subsequent phases to build upon the earlier efforts rather than duplicate them.

This paper will first look at the types of dictionary support used at each phase of the application life cycle and what could be done to use the dictionary for the interface between these phases. Then the paper will focus on how this might be implimented using DICTIONARY/3000. Following that will be a recap of specific tools that will be needed. Many of these do not exist today.

APPLICATION LIFE CYCLE--DICTIONARY SUPPORT

This author has chosen to break the application life cycle down into the following phases:

PLANNING
ANALYSIS
DESIGN
CODE
PRODUCTION
END-USER QUERY
MAINTENANCE
REPLACEMENT

This may not be the precise arrangement for any given development; but, they provide a common framework that the reader can tailor to his particular need. **ANALYSIS**

According to DeMarco, the most important product of this phase is the specification document. There are a number of ways to get there two of which (Holland and DeMarco) will be addressed here.

Holland's top down approach focuses on the processes occuring within the business. This approach is primarily used by Weyerhaeuser to define the logical business model. A given business is divided into the main FUNCTIONs needed to support that business. Each FUNCTION is divided into PROCESSes which are further divided into ACTIVITYs. At this point, the entity data requirements are defined for each ACTIVITY.

DeMarco's approach for "Structured Analysis and System Specification" is primarily concerned with DATA FLOWS and secondarily with the PROCESSES that modify the data within the DATA FLOW.

Both of these approaches are heavily graphics oriented. They are also iterative. Thus, there is a lot of maintenance to the graphic representation of the logical model being developed.

In the analysis stage the dictionary definitions and known attributes of the items (FUNCTION, PROCESS, ACTIVITY, ENTITY DATA FLOW, etc.) are continually being refined as the analyst gains additional knowledge regarding the project. Since the relationship of these items to each other as shown in the graphic representation is an attribute of the items, this relationship should be maintained in the dictionary. It then makes sense that the dictionary (or another tool making inquiry to the dictionary) would draw the graphic representation--be it the logical business

model, the data flow diagram (DFD), or the logical data structure diagram (DSD) or entity relationship diagrams.

Dictionary use may be complicated during this phase by the use of an analysis tool that supports one or more of the structured approaches to system analysis. If that is the case, a facility to transfer information bi-directionally between the analysis tool and the dictionary is a necessity.

DESIGN

This phase turns the logical model of the analysis phase into a physical model. Thus the DSD become databases and other files. The DFD is turned into a FUNCTIONAL CHART (STRUCTURE CHART, ORG CHART) which is refined until the result is functionally valid for the physical implementation that is envisioned. The STRUCTURED SPECIFICATION is not readily turned into a useful FUNCTIONAL CHART even when transaction and transform analysis are used. The output of the analysis phase is logically oriented and many of the physical aspects of the system are ignored. Also ignored are the control paths, the exception routines, etc., that the physical model must address.

The dictionary is updated with the additional physical attributes of the system as they become known. In addition, the normalized (hopefully) logical DSD is used to create the physical DSD which is also maintained in the dictionary. It would be advantageous if a tool could be used to make the first cut at the transformation from logical to physical data structure.

The FUNCTIONAL CHART that is one of the outputs of the design phase is a rather physical representation of the final system of programs and procedures. Thus the actual names of the routines, their relation to each other, and the data elements used (files accessed) are known. This information should be put into the DICTIONARY in a form that makes updating fairly easy. The charts could be one of the outputs of the DICTIONARY or some other tool that would make inquiry to the DICTIONARY.

CODE

While changes continue to be made in the design--and these should be updated in the DICTIONARY version of the STRUCTURED SPECIFICATIONS and FUNCTIONAL CHART--the major DICTIONARY support requirements for this phase include (but not limited to):

- Listing the detail specifications for procedures or modules the programmer is coding.
- Code generation for the table handling portions of the system.
- Code generation according to the detailed design specs.
- Creation and maintenance of language specific copylib or include files. It would be better if the individual language compilers would accept directives in the source code from which the compiler would determine what type of inquiry to make into the DICTIONARY and which information to copy from the DICTIONARY into the source.
- Compile stream generator. Most, if not all, of the information needed to compile a program (or the entire system) is in the DICTIONARY (or can be easily added). This would negate the headache of maintaining a seperate set of job streams for compilation.
- Continuing maintenance of the documentation regarding the logical and physical aspects of the system.

PRODUCTION

By this phase most of the DICTIONARY maintenance is done. There are several items that are of use to this phase:

- Batch job stream documentation. Rather than maintain an operator guide book that documents each step of a batch job the documentation should be in the dictionary and made accessible to the operator.
- On-line help facility within the DICTIONARY (or documented where it is maintained).
- User manuals maintained within the DICTIONARY (or documented where it is maintained).

The argument is that all information regarding the system should be documented in the DICTIONARY. This may not be possible for all types of information. In that case, effort should be made to document in the DICTIONARY where those additional items of information can be located. In addition, there should be a minimum of redundant information both within the DICTIONARY and between the DICTIONARY and the additional items of information.

MAINTENANCE

The DICTIONARY is now a repository of information to help the maintenance crew do its job. Inquiry can be made as to what 'things' are affected if changes are made to a specific 'thing'. This phase becomes a mini-life cycle for each change envisioned. The changes are walked through the STRUCTURE

SPECIFICATION to get the impact on the logical structure. Next the FUNCTIONAL CHART is modified to locate the changes on the physical structure. Finally, the related documentation entries are updated to reflect the change, and it is implemented using many of the concepts noted above.

REPLACEMENT

The application life-cycle has now come full circle. The DICTIONARY has one more major function before the replacement system takes over. It serves as the major source of information to be downloaded to the analysis tools. Also, many of the structures and other pieces of information/documentation can be salvaged. Like the phoenix rising from the ashes, the DICTIONARY of the new system is an outgrowth and metamorphosis of its previous self. The following table documents the major items that should be contained in the dictionary.

Planning

- Enterprise Model
  * Function
  * Process
  * Activity
  * Info. requirements
- Documents
- Entities
- Subject databases
- Business data elements

Analysis

- Logical/Functional Spec.
  * Data flows
  * Data processes
  * Logical files (logical DSD)
- Logical data elements

Design

- Physical Spec.
- Transactions
- Modules
- Physical file
- Records
- Screens
- Reports
- Physical data elements

Code

- Procedures
- Jobs
- Jobsteps
- Programs
- Processing data elements

Maintenance

- Additions, changes, and deletions to above specs. and documents
- +/- data element changes

The data elements hold all these documents together. They are the common thread pervading the entire life cycle. As such, it is essential to document their evolution from the conceptual 'Business Data Elements' to the quantified 'Physical Data Elements' and 'Processing Data Elements'.

DICTIONARY/3000 IMPLEMENTATION

Understanding possible methods to impliment the above ideas within DICTIONARY/3000 requires a careful investigation of what is available within the DICTIONARY. The ENTITY USAGE CHART (Figure 1) shows the conceptual arrangement of DICTIONARY/3000. (For the internal arrangement and physical layout see DICTIONARY/3000 Internal Structure--Figure 2; and DICTIONARY/3000 Schema--List 1.)

Just as DICTIONARY is the hub of the application life-cycle, the **ELEMENT** entity is the hub of the dictionary. The ELEMENT entity is oriented toward the physical data elements. Every attribute of ELEMENT has an HP defined usage--most of them tied rather tightly with the RAPID products. The TYPE attribute of ELEMENT can be left undefined (which means the other physical attributes are also undefined).

The **FILE** entity describes the different file structures that are available on the HP 3000. For some of these structures (such as IMAGE database and VPLS form files) the internal members are described as file types which indicate which elements are used and in what order they occur. Every attribute of the FILE entity has an HP defined meaning.

The **CLASS** entity defines the security for the ELEMENT and FILE entries. All security structures of IMAGE are supported. If there is no security defined at the FILE entry level for a particular data set, then the security for that data set is the resultant matrix of the securities for the data elements contained in that data set.

The TYPE attribute has no HP defined meaning--rather, whatever is meaningful to you. The field should be used to organize the security classes according to some classification of their usage. For example, the following types would be for the indicated usage: PROG - programatic access, the program is the end user rather than using a password on behalf of a user.

> The concept is used by the RAPID compilers and especially by INFORM. It doesn't matter what capability the user has, the program itself needs this access in order to do its thing.

MNGR - for the user who has ultimate responsability for the entire database. A program may even have this hard coded but the end result is that the program works on behalf and at the behest of the data base manager.

USER - level of access for some user. The CLASS-NAME would further define what user this password is for. There may even be programs set up for the user which have the password hard coded in them.

MANT - maintenance level. For the person or group that maintains a certain section of the data base. Used for fixups, corrections, etc. above and beyond the USER level. Indeed, this may be the same as MNGR for certain sections of the data base.

UTIL - a low level access for general purpose utilities to do structure checks, etc. These utilities would never change the database and would even have no predefined

xxxx - idea of the database structure. send ideas to the author.

The **GROUP** entity (aka INFORM GROUPS) defines the end user view for ad hoc reporting. It has the extra capability to indicate what file the associated elements reside in. It is only when the entry is related to $MENU or one of its children (grandchildren, etc.) that INFORM will include it in a menu of groups. When not in that structure, GROUP entries could be used to document other types of end user views. These would mainly be ordered lists of elements that reside on a report or other type of document. Remember that screens are documented in the FILE entries. The TYPE attribute is the only field for which HP has not defined a meaning. As noted under the CLASS discussion, the attribute could be used to indicate that a document was being documented. In addition, the TYPE could be used to classify the INFORM GROUPS. That will be left as an excercise for the reader.

LOCATION is little used (if used at all). Every field has an HP defined meaning--though CPU is not as HP oriented as ACCOUNT and GROUP. There is no TYPE field to help classify the LOCATION. If there had been, it could be used to indicate which LOCATION was SOURCE, OBJECT, DATA, USL, etc. Such cannot be done directly. A standard GROUP naming convention will help. This will be addressed further under the PROCEDURE discussion below.

There are two entities left (CATEGORY and PROCEDURE) for which many of the attributes have defined meanings. The usage of these is not well defined and, in many cases, not even attempted. It is in these that the structures mentioned in the Application Life Cycle section will be placed.

CATEGORY will document all the logical information. Things like the BUSINESS MODEL, DATA FLOW, and LOGICAL DATA STRUCTURE are prime candidates. Each entry will need a unique primary name that exists no other place within the CATEGORY structure of DICTIONARY. Thus a DATA FLOW member cannot have the same name as that of a BUSINESS MODEL member.

A more detailed look at CATEGORY shows these attributes:

image CATEGORY - a twenty (20) character field that is the primary name of the member entry. For the BUSINESS MODEL, the name is that of a FUNCTION, a PROCESS, an ACTIVITY, etc. This is upper case alpha-numeric.

CATEGORY-PARENT - the name of a CATEGORY which will serve as a parent in the relationship between two categories.

CATEGORY-CHILD - the name of a CATEGORY that is the child in the relationship between two categories.

CATEGORY-NAME - a fifty (50) character long title for the CATEGORY. This is descriptive of the CATEGORY entry without being part of the description. This is upper case alpha-numeric.

CATEGORY-RESP - a twenty (20) character alpha-numeric field that is the name of the person, department, or other that is responsible for the information represented by the meta-data.

CATEGORY-TYPE - a four (4) character upper case alpha-numeric field used to indicate the classification of this CATEGORY.

NOTE: The DICTIONARY/3000 command to create a CATEGORY is CREATE CATEGORY. To relate two categories together (as noted below) use RELATE CATEGORY. To associate an ELEMENT with a CATEGORY use ADD CATEGORY.

A standard set of CATEGORY-TYPE values are:

BUSINESS MODEL

BSPM - BUSINESS SYSTEM PLAN MODEL. This is the highest level of a BUSINESS MODEL. There is one of these for each business model in DICTIONARY. The lower levels consist of functions, processes, activities, etc.

FCNT - FUNCTION within the BSP. This is a child to a BSPM and parent to ACTN.

ACTN - ACTION within the FUNCTION. As with the FUNCTION, the CATEGORY named in this entry is of this type.

INFO - INFO REQUIREMENTS. This CATEGORY names an information requirement of the action to which this is a child.

The information requirements may be associated with the entries in the ELEMENT entity. This is done by the ADD CATEGORY command within DICTIONARY/3000.

LOGICAL DATA MODEL

LDSD - LOGICAL DATA STRUCTURE DIAGRAM. This is the highest level for the LDSD. The rest of the structure is made up of a table of files containing elements. This is the primary method of showing relational (ie, 3rd/4th Normal Form) data structures.

LDF - LOGICAL DATA FILE. These are children of the corresponding LDSD. They define the logical data files within the data structure.

LDEx - LOGICAL DATA ELEMENT. These are the children of the LDF's. The 'x' defines the type of logical element being defined.

P - This logical element is the primary key. There is only one primary key. It behaves the same as any of the secondary keys but has been chosen by the analyst/designer to serve as the primary way of accessing the data.

S - This logical element is a secondary key. There may be more than one secondary key; but, each is capable of being the identifier as is the primary key.

N - This logical element is not a key.

The data elements that make up each LDEx are associated to the appropriate category. This is the link back to the central hub.

SUBJECT DATA BASE

SDB - SUBJECT DATA BASE. This is the highest level for a given subject data base. It corresponds to the BASE type in the FILE entity.

SDF - SUBJECT DATA FILE. These are the files within the subject data base. They correspond to the AUTO, MAST, or DETL type entries in the FILE entity.

SDEx - SUBJECT DATA ELEMENT. This is the low level. It is similar to the low level of the LOGICAL DATA MODEL in that the rules for 'x' above also apply here with logical element changed to subject element. The data elements within ELEMENT that make up each SDEx are associated via the ADD CATEGORY command.

NOTE: At this point the question is raised--Why both a LOGICAL DATA MODEL and SUBJECT DATA BASE structures? The former is a part of an APPLICATION or BUSINESS SYSTEM; the later is for the entire business. Thus two more TYPES are:

APPL - This defines an APPLICATION. It has one each of:

1. LOGICAL DATA MODEL (LDSD).
2. DATA FLOW DIAGRAM (DFD)--yet to be defined.
3. FUNCTIONAL ORG CHART (FOC)--yet to be defined.
4. other organization documents defined by the user.

BUSS - This defines a BUSINESS. It has one or more of the following:

1. BUSINESS SYSTEM PLAN MODEL (BSPM). (Usually only one).
2. APPLICATION (APPL).

3. SUBJECT DATA BASE (SDB).

4. others not yet defined.

DATA FLOW DIAGRAM

This requires two parallel decompositions. One is for the actual data flows and the other is for the processes. The data flow will be documented in the CATEGORY entity. The process will be documented in the PROCEDURE entity (see below). The documentation for the process will describe how to link these two organizations together.

DFD - DATA FLOW DIAGRAM. The highest level for the data flow (or BUBBLE CHART) documentation within an APPLICATION (APPL).

DFP - DATA FLOW PIPE. The pipe that runs between the processes. This is a child of the DFD.

DFF - DATA FLOW FILE. A special pipe that is a data store; or, as Gane and Sarson state: "Data flows are data structures in motion; data stores are data structures at rest." This is a child of the DFD or of a DFDB.

DFDB - DATA FLOW DATA BASE. A special data store (DFF) that consists of many data stores. That is, a DFDB can have DFF's related to it. This is a child of the DFD.

DFE - DATA FLOW ELEMENT. A decomposition of the DFP, DFF, or DFDB. For DFF or DFDB one may choose to use the 'x' extender as is done for the LOGICAL DATA MODEL.

NOTE: The LOGICAL DATA MODEL is the outgrowth of the DATA FLOW structure. There will be many similarities.

PROCEDURE documents the programs, modules, and processes. The process half of the DATA FLOW DIAGRAM goes here. Also included are the FUNCTIONAL CHART as well as a description of the physical system as delivered to the user.

These attributes are available:

PROCEDURE - a twenty (20) character upper case alpha-numeric field that is the primary name of the entry.

PROCEDURE-PARENT - the name of a PROCEDURE which will serve as a parent in the relationship between two procedures.

PROCEDURE-CHILD - the name of a PROCEDURE that is the child in the relationship between two procedures.

PROCEDURE-NAME - a fifty (50) character upper case alpha-numeric long title for the PROCEDURE.

PROCEDURE-RESP - a twenty (20) character alpha-numeric field that is the name of the person, department, or other that is responsible for the information represented by the meta-data.

PROCEDURE-LANG - a ten (10) character upper case alpha-numeric field that holds the language code that the procedure is written in. These values need to be standardized across the HP environment.

PROCEDURE-TYPE - a four (4) character upper case alpha-numeric field used to indicate the classification of this PROCEDURE.

A standard set of PROCEDURE-TYPE values are:

BUSS - This defines a BUSINESS. The PROCEDURE name must match an identical CATEGORY name of the same type. It is used to hold those PROCEDURE entries together that belong to the business. There may be more that one business documented in DICTIONARY.

APPL - It has the same function for the PROCEDURE as in CATEGORY. There must be a common name of this type in each. There may be many of these for a given

business. Attached to this are the process half of the DFD, the FUNCTIONAL CHART, and the SYSTEM CHART.

DATA FLOW DIAGRAM

DFD - DATA FLOW DIAGRAM. The PROCEUDRE name for this entry must have a corresponding entry for a CATEGORY of like type.

DFPL - DATA FLOW PROCESS LEVEL. The PROCEDURE is the name of a process bubble. The first level DFPL are related to the DFD. Lower level DFPL are related to higher level DFPL.

DFIN - DATA FLOW INPUT. The PROCEDURE name is a name of a CATEGORY entry of type DFP, DFDB, or DFF. This names the data flow that is input to the DFPL to which this is related. The DFIN cannot have any children within the PROCEDURE entries. This is because it is defined within the CATEGORY section of DICTIONARY/3000.

DFOT - DATA FLOW OUTPUT. The PROCEDURE name is a name of a CATEGORY entry of type DFP, DFDB or DFF. This names the data flow that is output from the DFPL to which it is related. Like the DFIN, the DFOT cannot have any children.

DFIO - DATA FLOW INPUT/OUTPUT. The PROCEDURE name is the name of a CATEGORY entry of type DFP, DFDB or DFF. This names the data flow that passes both into and out of (named the same) the DFPL. As with the DFIN and DFOT, the DFIO cannot have any children.

DFEX - DATA FLOW EXTERNAL. This PROCEDURE is either a source or a sink or both. It is used in place of a DFPL when the PROCESS is in reality a SINK or SOURCE. This may have DFIN, DFOT, or DFIO (not likely) as children. They define the data flows to and from the external process (ie, SOURCE or SINK).

FUNCTIONAL CHART

The first level cut of the FUNCTIONAL CHART comes from transform and transaction analysis of the DFD. Thereafter, the designers modify it until the results are acceptable. There may be more than one FUNCTIONAL CHART for an APPLICATION.

FCHT - FUNCTIONAL CHART. The PROCEDURE is the name of the high level module (box) in the FUNCTIONAL CHART.

FCMD - FUNCTIONAL CHART MODULE. The PROCEDURE is the name of the module (any level box) within the FUNCTIONAL CHART. This is a child of the FCHT or another FCMD.

FCDN - FUNCTIONAL CHART DOWNFLOW. The PROCUDURE is the name of the data flow down to the FCMD of which this is a child. (Complicated sounding isn't it! Relax, if your FUNCTIONAL CHART doesn't show the data flow between module levels then don't use it.) This FCDN is a child of an FCMD. That FCMD receives data down from its parent. This FCDN is the name of that flow. Associated to this FCDN are ELEMENTS.

FCUP - FUNCTIONAL CHART UPFLOW. The PROCEDURE is the name of the data flow up from the FCMD of which this is a child. That is, this FCUP is a child of an FCMD. That FCMD sends data back to its parent. This FCUP is the name of that flow. Associated to this FCUP are ELEMENTS (use the ADD PROCEDURE command and give this PROCEDURE name when prompted).

THE SYSTEM

Whatever this is called, it is the finished product of jobstreams, programs, subroutines, etc. that is delivered to the user.

SYS - SYSTEM. This PROCEDURE is the name of the system. It is a child of the BUSS procedure.

SUBS - SUB-SYSTEM. This PROCEDURE is the name of one of the sub-systems within the system.

FCTN - FUNCTION. This PROCEDURE is a function within the SUBS.

JOBS - JOB STREAM. This PROCEDURE is a job stream within the FCTN.

PROG - PROGRAM. This PROCEDURE is a program within either the on-line portion of the FCTN (ie, FCTN procedure is the parent) or within the job stream (JOBS procedure is the parent).

SUBP - SUBPROGRAM. This PROCEDURE is a sub-program within a PROG. This is written and is compiled directly into the same USL as is the PROG.

SLRT - SL ROUTINE (non-system SL). This PROCEDURE is a sub-program that is placed into the account SL.

SLSY - SL SYSTEM ROUTINE. This PROCEDURE is a system level routine.

FILE - FILE. This PROCEDURE is a file (found in the FILE entity entries) used by the PROG, SUBP, or SLRT. The actual ELEMENT entries used by the parent will be associated with the appropriate file. If there is no file, then the elements will be associated with the PROG, SUBP, etc.

This last may seem like a duplication of what the FILE entity entries show. To some extent it is in that all entries here are in the FILE entries also. The difference is that the procedure does not use all the elements within a file. This allows the DBA to document which elements from what files are actually used. These are to be considered as the procedure's sub-schema.

DESCRIPTIONS

The DESCRIPTION entries for each entity, relation, and association have different uses.

The description entry made during the CREATE command of DICTDBM describes the entry in its general case. Documentation that applies across the board to the entry is to be entered here. This description is output during the LIST, DISPLAY, and REPORT commands of DICTDBM.

The description made during the RELATE command describes the relation between the two occurences of the entity. Anything specific to the relation is to be documented at that time.

The description made during the ADD command describes the association between the two different entities. Document the specific information that relates to this association.

This may take some experimentation to understand what is happening. Basically, each entry can have three (3) descriptions. Some can have more. For example, the FILE entity has the following:

CREATE FILE - this description gives information about the file in general. What its purpose is, ...

RELATE FILE - description gives information about the relation between the two files. Any special considerations when this data set is in this base (or this form is in this view file) are documented at this point.

ADD FILE - describes the association between the element and the file. Any special constraints for the element when it is in the file.

ADD FILE-LOC -describes the association between the file and the location.

ADD FILE-CLASS - and the documentation of the association of this file with that class.

SUMMARY

The DICTIONARY must become the interface between the many phases of the application life cycle. With the information internally maintained so that the graphic representation can be produced the DBA is able to utilize tools to insure a consistent description of the developing system.

Likewise, the production support people have at their fingertips the detailed information regarding the system from initial conception until final delivery. The run time manuals are stored in the DICTIONARY to be retrieved on-line as the need arises. The end user has access to some help facility to gain the information that normally would be hid on page x-y of the user manual. Described above is a mechanism to start down that road. With newer and better tools available, the long victorian novels

that concern Tom De Marco will become a thing of the past. The "STRUCTURED ANALYSIS and SYSTEM SPECIFICATION" (Tom De Marco) will become facility of the DICTIONARY. When the programmer is ready to code, all of the supporting information, formats, files, etc. will be in the DICTIONARY ready to be utilized.

My many thanks to members of the Data Administration unit who gave their input. Especially to Larry Rolstad who must do the same for the IBM dictionary.

BIBLIOGRAPHY

Date, CJ. **An Intorduction to Database Systems** Vol 1, 3rd ed. Addison-Wesley Publishing Co., 1981.

De Marco, Tom. **Structured Analysis and System Specification** Yourdon, 1979.

Dowling, Jim. "RAPID is a Relative Term" **MONTREAL PROCEEDINGS**, 1983.

Larson, Orland. "Software Prototyping: Today's Approach to Information Systems Design and Development", **MONTREAL PROCEEDINGS**, 1983.

Puckering, Gary, "Data Dictionaries--A New Era", **MONTREAL PROCEEDINGS**, 1983.

```
**** FIGURE SMBTXT-1 GOES ON THIS PAGE
**** FIGURE SMBTXT-2 GOES ON THIS PAGE
DICTIONARY/3000 Schema--List 1
$CONTROL BLOCKMAX=1024

BEGIN DATA BASE DICT;

PASSWORDS:
     1  HPPRGUSE;   <<HP programatic use password>>
     2  MANAGER;    <<MANAGER access--not realy total; but close>>
     3  PROGRAMR;   <<PROGRAMMER access>>
     4  INFORM;     <<INFORM group creator/maintainer>>
     5  DOCUMENT;   <<DOCUMENTATION access for the documentarian>>
     6  REPORT;     <<REPORT access for those who want reports>>
                    <<about DICTIONARY contents.  No relation >>
                    <<to usage of REPORT/3000.                >>

ITEMS:
     CATEGORY,              U20       (6/1,2,3,4,5);
     CATEGORY-CHILD,        U20       (6/1,2,3,4,5);
     CATEGORY-NAME,         U50       (6/1,2,3,4,5);
     CATEGORY-PARENT,       U20       (6/1,2,3,4,5);
     CATEGORY-RESP,         X20       (6/1,2,3,4,5);
     CATEGORY-TYPE,         U4        (6/1,2,3,4,5);
     CLASS,                 I1        (4,5,6/1,2,3);
     CLASS-NAME,            U50       (4,5,6/1,2,3);
     CLASS-PASSWORD,        X8        (3/1,2);
     CLASS-RESP,            X20       (4,5,6/1,2,3);
     CLASS-TYPE,            U4        (4,5,6/1,2,3);
     CONTROL-KEY,           I1        (6/1,2,3,4,5);
     CONTROL-OPTIONS,       U30       (6/1,2,3,4,5);
     DATE-CHANGE,           U6        (6/1,2,3,4,5);
     DATE-CREATE,           U6        (6/1,2,3,4,5);
     DESCRIPTION-KEY,       I2        (6/1,2,3,4,5);
     DESCRIPTION-LINE,      X50       (6/1,2,3,4,5);
     ELEMENT,               U20       (6/1,2,3,4,5);
     ELEMENT-ACCESS,        U2        (4,5,6/1,2,3);
     ELEMENT-ALIAS,         U20       (5,6/1,2,3,4);
     ELEMENT-ALIAS-C,       U60       (6/1,2,3,4,5);
```

```
        ELEMENT-ALIAS-P,    U30       (6/1,2,3,4,5);
        ELEMENT-CHILD,      U20       (4,5,6/1,2,3);
        ELEMENT-COUNT,      I1        (4,5,6/1,2,3);
        ELEMENT-DEC,        I1        (4,5,6/1,2,3);
        ELEMENT-DISPLAY,    I1        (5,6/1,2,3,4);
        ELEMENT-EDIT,       X30       (4,5,6/1,2,3);
        ELEMENT-ENTRY,      X30       (4,5,6/1,2,3);
        ELEMENT-HEADING,    X30       (4,5,6/1,2,3);
        ELEMENT-KEY,        I2        (6/1,2,3,4,5);
        ELEMENT-LENGTH,     I1        (4,5,6/1,2,3);
        ELEMENT-NAME,       U50       (4,5,6/1,2,3);
        ELEMENT-PARENT,     U20       (4,5,6/1,2,3);
        ELEMENT-POSITION,   I1        (4,5,6/1,2,3);
        ELEMENT-PRIMARY,    I1        (4,5,6/1,2,3);
        ELEMENT-RESP,       X20       (4,5,6/1,2,3);
        ELEMENT-SIZE,       I1        (4,5,6/1,2,3);
        ELEMENT-TYPE,       U2        (4,5,6/1,2,3);
        ELEMENT-UNITS,      X10       (4,5,6/1,2,3);
        FILE,               U20       (6/1,2,3,4,5);
        FILE-ACCESS,        U2        (4,5,6/1,2,3);
        FILE-ALIAS,         U8        (6/1,2,3,4,5);
        FILE-ALIAS-F,       U16       (4,5,6/1,2,3);
        FILE-BLOCK,         I1        (4,5,6/1,2,3);
        FILE-CHILD,         U20       (4,5,6/1,2,3);
        FILE-KEY,           I2        (6/1,2,3,4,5);
        FILE-NAME,          U50       (4,5,6/1,2,3);
        FILE-PARENT,        U20       (4,5,6/1,2,3);
        FILE-PARENT-KEY,    I2        (5,6/1,2,3,4);
        FILE-RESP,          X20       (4,5,6/1,2,3);
        FILE-SIZE,          I2        (6/1,2,3,4,5);
        FILE-TYPE,          U4        (4,5,6/1,2,3);
        GROUP,              U20       (5,6/1,2,3,4);
        GROUP-CHILD,        U20       (5,6/1,2,3,4);
        GROUP-NAME,         U50       (5,6/1,2,3,4);
        GROUP-PARENT,       U20       (5,6/1,2,3,4);
        GROUP-RESP,         X20       (5,6/1,2,3,4);
        GROUP-TYPE,         U4        (5,6/1,2,3,4);
        IDENTITY-CHANGE,    U8        (6/1,2,3,4,5);
        IDENTITY-CREATE,    U8        (6/1,2,3,4,5);
        LINK-VALUE,         I1        (5,6/1,2,3,4);
        LOCATION,           U20       (6/1,2,3,4,5);
        LOCATION-ACCOUNT,   U8        (6/1,2,3,4,5);
        LOCATION-CPU,       U8        (6/1,2,3,4,5);
        LOCATION-GROUP,     U8        (6/1,2,3,4,5);
        LOCATION-NAME,      U50       (6/1,2,3,4,5);
        POSITION,           K2        (6/1,2,3,4,5);
        PROCEDURE,          U20       (6/1,2,3,4,5);
        PROCEDURE-ALIAS,    U8        (6/1,2,3,4,5);
        PROCEDURE-CHILD,    U20       (6/1,2,3,4,5);
        PROCEDURE-LANG,     U10       (6/1,2,3,4,5);
        PROCEDURE-NAME,     U50       (6/1,2,3,4,5);
        PROCEDURE-PARENT,   U20       (6/1,2,3,4,5);
        PROCEDURE-RESP,     X20       (6/1,2,3,4,5);
        PROCEDURE-TYPE,     U4        (6/1,2,3,4,5);
        REPORT,             U6        (/1);
        REPORT-LOC,         U20       (/1);
        REPORT-NAME,        U50       (/1);

    SETS:

    NAME:   DATA-ELEMENT,   MANUAL    (4,5,6/1,2,3);
    ENTRY:  ELEMENT         (8),
            ELEMENT-NAME,
            ELEMENT-TYPE,
            ELEMENT-SIZE,
            ELEMENT-DEC,
            ELEMENT-LENGTH,
```

```
          ELEMENT-COUNT,
          ELEMENT-UNITS,
          ELEMENT-RESP,
          ELEMENT-HEADING,
          ELEMENT-ENTRY,
          ELEMENT-EDIT,
          DATE-CREATE,
          DATE-CHANGE,
          IDENTITY-CREATE,
          IDENTITY-CHANGE,
          DESCRIPTION-KEY;
    CAPACITY: 303;

    NAME:   DATA-FILE,        MANUAL    (4,5,6/1,2,3);
    ENTRY:  FILE              (6),
          FILE-NAME,
          FILE-TYPE,
          FILE-RESP,
          DATE-CREATE,
          DATE-CHANGE,
          IDENTITY-CREATE,
          IDENTITY-CHANGE,
          DESCRIPTION-KEY;
    CAPACITY: 151;

    NAME:   DATA-PROCEDURE,   MANUAL    (6/1,2,3,4,5);
    ENTRY:  PROCEDURE         (4),
          PROCEDURE-LANG,
          PROCEDURE-NAME,
          PROCEDURE-RESP,
          PROCEDURE-TYPE,
          DATE-CREATE,
          DATE-CHANGE,
          IDENTITY-CREATE,
          IDENTITY-CHANGE,
          DESCRIPTION-KEY;
    CAPACITY: 11;

    NAME:   DATA-CATEGORY,    MANUAL    (6/1,2,3,4,5);
    ENTRY:  CATEGORY          (3),
          CATEGORY-NAME,
          CATEGORY-TYPE,
          CATEGORY-RESP,
          DATE-CREATE,
          DATE-CHANGE,
          IDENTITY-CREATE,
          IDENTITY-CHANGE,
          DESCRIPTION-KEY;
    CAPACITY: 101;

    NAME:   DATA-GROUP,       MANUAL    (5,6/1,2,3,4);
    ENTRY:  GROUP             (3),
          GROUP-NAME,
          GROUP-TYPE,
          GROUP-RESP,
          DATE-CREATE,
          DATE-CHANGE,
          IDENTITY-CREATE,
          IDENTITY-CHANGE,
          DESCRIPTION-KEY;
    CAPACITY: 11;

    NAME:   DATA-CLASS,       MANUAL    (3,4,5,6/1,2);
    ENTRY:  CLASS             (2),
          CLASS-NAME,
          CLASS-TYPE,
          CLASS-PASSWORD,
```

```
        CLASS-RESP,
        FILE-KEY,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY;
CAPACITY: 67;

NAME:   DATA-LOCATION,   MANUAL    (6/1,2,3,4,5);
ENTRY:  LOCATION        (2),
        LOCATION-NAME,
        LOCATION-GROUP,
        LOCATION-ACCOUNT,
        LOCATION-CPU,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY;
CAPACITY: 11;

NAME:   DIC-CONTROL,     MANUAL    (6/1,2,3,4,5);
ENTRY:  CONTROL-KEY     (0),
        DESCRIPTION-KEY,
        FILE-KEY,
        ELEMENT-KEY,
        CONTROL-OPTIONS,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE;
CAPACITY: 1;

NAME:   LINK-FILE,       AUTOMATIC (6/1,2,3,4,5);
ENTRY:  FILE-KEY        (1);
CAPACITY: 101;

NAME:   LINK-ELEMENT,    AUTOMATIC (6/1,2,3,4,5);
ENTRY:  ELEMENT-KEY     (1);
CAPACITY: 79;

NAME:   LINK-DESCRIPTION,AUTOMATIC (6/1,2,3,4,5);
ENTRY:  DESCRIPTION-KEY (1);
CAPACITY: 611;

NAME:   DATA-REPORTLOC,  AUTOMATIC (/1);
ENTRY:  REPORT-LOC      (1);
CAPACITY: 11;

NAME:   ELEMENT-ELEMENT, DETAIL    (4,5,6/1,2,3);
ENTRY:  ELEMENT-PARENT  ( DATA-ELEMENT    (POSITION        )),
        ELEMENT-CHILD   (!DATA-ELEMENT    ),
        ELEMENT-POSITION,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 11;

NAME:   FILE-FILE,       DETAIL    (4,5,6/1,2,3);
ENTRY:  FILE-PARENT     ( DATA-FILE       (POSITION        )),
        FILE-CHILD      (!DATA-FILE       ),
        FILE-ALIAS-F,
        FILE-SIZE,
```

```
        FILE-BLOCK,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 150;

NAME:   PROCEDURE-PROCED,DETAIL    (6/1,2,3,4,5);
ENTRY:  PROCEDURE-PARENT( DATA-PROCEDURE  (POSITION        )),
        PROCEDURE-CHILD (!DATA-PROCEDURE  ),
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 11;

NAME:   CATEGORY-CATEGOR,DETAIL    (6/1,2,3,4,5);
ENTRY:  CATEGORY-PARENT ( DATA-CATEGORY   (POSITION        )),
        CATEGORY-CHILD  (!DATA-CATEGORY   ),
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 150;

NAME:   GROUP-GROUP,     DETAIL    (5,6/1,2,3,4);
ENTRY:  GROUP-PARENT    ( DATA-GROUP      (POSITION        )),
        GROUP-CHILD     (!DATA-GROUP      ),
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 11;

NAME:   FILE-ELEMENT,    DETAIL    (4,5,6/1,2,3);
ENTRY:  FILE             ( DATA-FILE       (POSITION        )),
        ELEMENT          (!DATA-ELEMENT    ),
        ELEMENT-ALIAS,
        FILE-KEY,
        ELEMENT-KEY,
        ELEMENT-PRIMARY,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 600;

NAME:   FILE-PATH,       DETAIL    (6/1,2,3,4,5);
ENTRY:  FILE            (!DATA-FILE       ),
        FILE-KEY        ( LINK-FILE       );
CAPACITY: 100;

NAME:   FILE-SORT,       DETAIL    (6/1,2,3,4,5);
ENTRY:  ELEMENT         (!DATA-ELEMENT    ),
        ELEMENT-KEY     ( LINK-ELEMENT    );
CAPACITY: 80;
```

```
NAME:    PROCEDURE-ELEMEN,DETAIL    (6/1,2,3,4,5);
ENTRY:   PROCEDURE        ( DATA-PROCEDURE  (POSITION        )),
         ELEMENT          (!DATA-ELEMENT    ),
         ELEMENT-ALIAS-P,
         DATE-CREATE,
         DATE-CHANGE,
         IDENTITY-CREATE,
         IDENTITY-CHANGE,
         DESCRIPTION-KEY,
         POSITION;
CAPACITY: 11;

NAME:    CATEGORY-ELEMENT,DETAIL    (6/1,2,3,4,5);
ENTRY:   CATEGORY         ( DATA-CATEGORY   (POSITION        )),
         ELEMENT          (!DATA-ELEMENT    ),
         ELEMENT-ALIAS-C,
         DATE-CREATE,
         DATE-CHANGE,
         IDENTITY-CREATE,
         IDENTITY-CHANGE,
         DESCRIPTION-KEY,
         POSITION;
CAPACITY: 200;

NAME:    GROUP-ELEMENT,   DETAIL    (5,6/1,2,3,4);
ENTRY:   GROUP            ( DATA-GROUP      (POSITION        )),
         ELEMENT          (!DATA-ELEMENT    ),
         ELEMENT-ALIAS,
         FILE-KEY,
         FILE-PARENT-KEY,
         LINK-VALUE,
         ELEMENT-DISPLAY,
         DATE-CREATE,
         DATE-CHANGE,
         IDENTITY-CREATE,
         IDENTITY-CHANGE,
         DESCRIPTION-KEY,
         POSITION;
CAPACITY: 11;

NAME:    CLASS-ELEMENT,   DETAIL    (4,5,6/1,2,3);
ENTRY:   CLASS            ( DATA-CLASS      (POSITION        )),
         ELEMENT          (!DATA-ELEMENT    ),
         ELEMENT-ACCESS,
         DATE-CREATE,
         DATE-CHANGE,
         IDENTITY-CREATE,
         IDENTITY-CHANGE,
         DESCRIPTION-KEY,
         POSITION;
CAPACITY: 800;

NAME:    CLASS-FILE,      DETAIL    (4,5,6/1,2,3);
ENTRY:   CLASS            ( DATA-CLASS      (POSITION        )),
         FILE             (!DATA-FILE       ),
         FILE-ACCESS,
         DATE-CREATE,
         DATE-CHANGE,
         IDENTITY-CREATE,
         IDENTITY-CHANGE,
         DESCRIPTION-KEY,
         POSITION;
CAPACITY: 11;

NAME:    FILE-LOCATION,   DETAIL    (6/1,2,3,4,5);
ENTRY:   LOCATION         ( DATA-LOCATION   (POSITION        )),
         FILE             (!DATA-FILE       ),
```

```
        FILE-ALIAS,
        FILE-SIZE,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 11;

NAME:   PROCEDURE-LOCATI,DETAIL    (6/1,2,3,4,5);
ENTRY:  LOCATION        ( DATA-LOCATION   (POSITION        )),
        PROCEDURE       (!DATA-PROCEDURE  ),
        PROCEDURE-ALIAS,
        DATE-CREATE,
        DATE-CHANGE,
        IDENTITY-CREATE,
        IDENTITY-CHANGE,
        DESCRIPTION-KEY,
        POSITION;
CAPACITY: 11;

NAME:   DESCRIPTION-TEXT,DETAIL    (6/1,2,3,4,5);
ENTRY:  DESCRIPTION-KEY (!LINK-DESCRIPTION(POSITION        )),
        DESCRIPTION-LINE,
        POSITION;
CAPACITY: 1000;

NAME:   REPORT-LIST,     DETAIL    (6/1,2,3,4,5);
ENTRY:  REPORT-LOC      (!DATA-REPORTLOC  (REPORT          )),
        REPORT,
        REPORT-NAME,
        DATE-CREATE,
        IDENTITY-CREATE;
CAPACITY: 11;

END.
```

Hub of Systems Development and Documentation

Stephen M. Butler Weyerhaeuser Company

BIOGRAPHY

Stephen M. Butler is the Sr. Technical Specialist for the HP3000 within the I/S Data Administration unit of Weyerhaeuser Company. His area of expertise for the company is with IMAGE and the RAPID products. He has been with Weyerhaeuser since July, 1981.

Steve has six (6) years of hospital application development experience using both a Honeywell 115 (1 yr) and the HP 3000 (5 yrs). He was the Director of Data Processing during the last three (3) of those years.

In 1975, he obtained a BS degree in chemistry with very strong minors in mathematics and physics.

Steve is married and has three (3) children between the ages of two (last October) and five (last March).

PRIVILEGED MODE -- HOW TO USE IT SAFELY AND EFFECTIVELY

With Practical Applications

by Joseph C. Felix and Chris Hauck
Software Design Specialists
Educational Computer Systems, Inc.

INTRODUCTION

Privileged mode on the HP3000 is a concept which is often misunderstood by HP3000 users. It's the subject of many conversations and heated arguments. Many people are curious about PM but avoid it out of fear.

The authors feel that when used properly, privileged mode can be a very powerful tool. We openly admit that we don't know everything thing there is to know about privileged mode and MPE. We do feel, however, that countless hours of heuristic programming with a System Tables Manual in one hand and a cold load tape in the other, have given us valuable knowledge which we are only too glad to share.

This paper is intended to serve as a springboard for the experienced programmer who wishes to start experimenting with PM. We also hope to give some new ideas to PM gurus in return for the techniques we have "borrowed" from them.

The paper does touch on some internal MPE tables. Our intent is not to describe the tables in depth; to do so would be a monumental task. We, instead, will explain relationships between tables and describe how to access the tables. Anyone serious about privileged mode programming MUST obtain a MPE System Tables Manual to effectively make use of the techniques and information we supply here.

All information here is accurate to the best of our knowledge. All examples have been tested and they do run on the Q-MIT level C release of MPE. Of course, the authors assume no liability for consequences arising from or out of the use of information contained herein.

WHAT IS PRIVILEGED MODE?

Hewlett-Packard defines privileged mode as follows:

Privileged mode is characterized by the ability to execute privileged instructions and to call segments which have been declared UNCALLABLE.

The normal mode of operation on the HP3000 is called user mode. In order for a program running in user mode to do anything requiring privileged mode (such as input/output), it must call a non-privileged procedure or intrinsic which in turn calls the privileged routines necessary to complete the task. The person running in privileged mode can skip this intermediate step and DIRECTLY call the procedures or execute the machine instructions necessary.

Privileged mode is an attribute of accounts, groups, users, and code segments. In order for a user of group to possess privileged mode capability, the account must have it. Similarly, in order for a privileged program to run, it must reside on a group with PM capability.

WHY USE PRIVILEGED MODE?

Most people realize that one slip while running in privileged mode can destroy file integrity or crash the system, and for them, that's reason enough to avoid PM. However, there are many applications in which privileged mode can be put to good use.

1. It lets you access files with a negative file code - specifically, IMAGE data bases. Hence, you need PM (or at least OP capability) if you want to :STORE or :RESTORE a data base. Sure, DBSTORE works just as well, unless you want to store multiple data bases. Almost all data base utility programs use privileged mode.

2. Privileged mode is extremely useful for overcoming MPE bugs or shortcomings. Have you ever wanted to have a UDC repeat itself? or switch to a different group without logging on again? It's possible; use privileged mode. How about a utility program to show you all users on the system who are accessing a particular file or show all files currently accessed by a particular user? Did you ever want to see your temp files displayed in a :LISTF,2 format rather than that printed by LISTEQ2.PUB.SYS?

3. Another reason for using privileged mode is that it lets you take advantage of "short-cuts" built into MPE by HP for their convenience. Several intrinsics operate differently when called in privileged mode. The privileged user can use machine instructions not available to others for transferring data to and from any data segment on the system. Complex input/output routines become extremely simple. Read on -- we'll show you.

HOW TO IMPLEMENT PRIVILEGED MODE

Before getting into specifics, it's important to know exactly how to use privileged mode. There are basically three different ways of implementing privileged mode.

1. IN A PRIVILEGED PROGRAM -- You type in the program using EDITOR, and compile it as usual. The program must be :PREPped with the parameter CAP=PM in addition to any other capabilities you desire. In order to :RUN the program, you must :SAVE it on a group which has PM capability. If you run it as $OLDPASS or a TEMP file, then the user you are logged on as must have PM.

2. AS A PROCEDURE IN AN SL -- You may choose to write your routine as a procedure or function, omitting the outer block. Compile your procedure as usual, then use the :SEGMENTER to add it to an SL on a group having PM capabilty.

3. DEBUG -- Another way to use privileged mode is in privileged DEBUG. This is often overlooked by people, but is frequently the best way to make small changes, or spot check a system table. Privileged mode DEBUG can be entered by any user posessing PM capabililty simply by typing the :DEBUG command. Once in PM DEBUG, the inexperienced user should be cautious; it is very easy to crash the system while in PM DEBUG.

LANGUAGE CONSIDERATIONS

Once you understand some of the possibilities of using privileged mode, you must choose a language for your program. A program for accessing privileged files can be written in almost any language supported on the HP3000. However, we don't recommend using fourth generation languages, or RPG, or BASIC, or COBOL 68. These languages are awkward to use when writing programs that involve calling intrinsics. COBOL II, FORTRAN, and PASCAL make such programs easier, but we find SPL to be the best choice. SPL is the "native language" of the HP3000. Since the MPE operating system is written in SPL, anything that can be done on the HP3000 can be done in SPL. This is not necessarily true of other languages. SPL provides the programmer with the ability to call intrinsics easily, the ability to write in machine language, and the ability to access absolute memory locations. Since SPL is not taught at many universities, it's hard to find SPL programmers. However, SPL is an easy language to learn and anyone with a fairly decent programming background should have no trouble mastering it. Because of SPL's structure, it makes an excellent language for presenting examples. Anyone familiar with PASCAL, FORTRAN, or COBOL will have little trouble understanding a well- documented SPL example.

WRITING SAFE PM PROGRAMS

We've compiled a list of things to keep in mind when writing privileged mode programs or procedures. Most of these apply to privileged mode programs which access system tables, but many are applicable to programs using privileged files and other types of PM applications.

If you've ever wondered about the "crash potential" of a particular PM program, these guidelines can serve as faily decent criteria for evaluation of a PM program or procedure.

1. TEST YOUR PLANNED PROGRAM WITH PM DEBUG FIRST, IF POSSIBLE. If you're planning to write a program that prints entries from an MPE table, or goes in and modifies one, it's a good idea to try to do it in DEBUG first. By checking your approach step by step, you can spot any flaws in locating the information you desire. Privileged Mode DEBUG is not foolproof, however. Be sure your system manager knows of your activities and by all means, do your experimentation on an unloaded system.

2. START BY WRITING READ-ONLY PM PROGRAMS, NOT PROGRAMS WHICH GO IN AND CHANGE TABLES. We feel that one of the shortcomings of MPE is the lack of a "Read Only Privileged Mode" capability. The way MPE was designed, PM capability gives the user the ability to access AND CHANGE anything in the operating system. This makes it all too easy to accidentally destroy MPE. Though it may seem obvious, we recommend that the novice PM user stick with programs which read tables and PM files, and stay away from programs which change them.

3. USE TEMPORARILY PRIVILEGED PROGRAMS WHENEVER POSSIBLE. It's always a good idea to minimize the time your program spends running in privileged mode. For this reason, you should use the GETPRIVMODE and GETUSERMODE intrinsics to bounce in and out of privileged mode. Stay in user mode as long as possible, then GETPRIVMODE, do your thing, then GETUSERMODE.

4. ANTICIPATE POTENTIAL BOUNDS VIOLATION SITUATIONS. Since MPE has been in operation, bounds checking for programs running in privileged mode has been minimal (although it's rumored that the Series 64 now does PM bounds checking). Operations which produce bounds violations in user mode execute OK in privileged mode, sometimes destroying data in undesired locations. To prevent this, your program should check DST sizes, and array boundaries, etc.

5. DST CHECKING. Data Segment number 2 is a table containing information about all data segments in the system. It's always a smart move to check this data segment to be sure you're not going to try to access a nonexistent data segment or read past the end of an existing one.

6. NEVER :RELEASE A PM PROGRAM. Any released file can be written to by any user on the system. A released PM program could be potential disaster. Anyone can change this program to do what they want it to, or get into PM debug. We recommend the use of lockwords to help you keep a constant watch on the security of your privileged programs.

7. DISABLE TRAPS WHILE EXECUTING PRIVILEGED CODE. To prevent unwanted interrupts while your privileged program is running, disable arithmetic traps, user interrupts, and control Y handling Interrupts are not normally a problem, as long as your program is prepared to handle them. To keep programs simple and straightforward, we recommend disabling traps.

8. DISABLE THE BREAK KEY. The last thing you want to have happen in your privileged program is to have a user press break and :ABORT in the middle of a table change. By all means, disable the break key; especially if your program is going to do any table or file updating.

9. CHECK THE MPE VERSION OR SYSTEM CONFIGURATION FOR DEPENDANT FEATURES. If you know for sure that your program works on one particular version of MPE, you might want to check for that version before you have your program do its thing. This is particularly important if you suspect that the program may not run on future releases of MPE. Locations %114-%116 in DST #5 (System Global Area) contain the current update, fix level, and version of MPE.

10. YOU MAY WANT TO CHECK THE CPU TYPE. Use the privileged instruction PCN (Push CPU Number) or the callable procedure THISCPU. We recommend THISCPU for the simple fact that PM is not needed to call it. The return values are:

CPU TYPE	PCN RETURNS	THISCPU
II	1	1
30,33	8	2
III	2	3
40,44	3	4
64	4	5

11. USE ENTRY SIZE VALUES IN TABLE HEADERS OR ENTRY NUMBER ZERO OF MOST TABLES. The idea here is to hard-code as little information as possible about system tables within your program. Many tables contain information about themselves such as the number of entries, entry size, etc. By using the values contained in the tables, your program is more likely to run on future releases of MPE. If HP changes the table size, your program will pick it up at run time, and still be OK.

12. KEEP GOOD DOCUMENTATION. The importance of this can not be overstressed. Major items to document are: A. Assumptions your program makes, if any. 1. DSTs of particular tables 2. Sizes of particular tables 3. Relationships between tables B. DEBUG test procedures -- How can you verify the operation of your program using PM DEBUG C. Table names and entries your program uses D. Table names and entries your program changes (if any)

Also, keep your documentation up to date. You'll need it in the next step.

13. Re-evaluate your program after major releases of MPE. HP often restructures tables from one release of MPE to another. Past experience has shown that though the tables may change and data may move around, the information you need is still in there somewhere. Your PM program should require little modification, if any, to go from one release of MPE to another. However, since there is always the possibility of table changes, we recommend that you do take the time to analyze your program to prevent disastrous results.

14. USE SIRS AND SETCRITICAL WHEN APPROPRIATE. MPE handles table contention through the use of SIRs. Certain tables should only be accessed after you've locked (obtained) the corresponding SIR. The MPE System Tables Manual has a list of tables and their corresponding SIRs. It's also important that you get SIRs in the correct order. Never attempt to get a SIR with a higher number than any SIR you already hold. A SIR deadlock could result.

A program running in privileged mode can also call SETCRITICAL and RESETCRITICAL. Once a procedure is "CRITICAL", the system will continue to execute this procedure only, until it calls RESETCRITICAL. This is another way to make sure a table has not changed between successive reads or between a read and an update. Be careful, if your process aborts for any reason while it is CRITICAL, a system failure 311 will result. Likewise, if you get a SIR, you better release it before termination or a system failure 314 will ensue.

MPE TABLES

As mentioned in the introduction, we intend to provide a brief introduction to some MPE tables. Table descriptions are kept purposely brief. Our goal is to acquaint you with some important MPE tables, show useful examples, and encourage further reading on your part.

The tables we'll introduce are divided into three major classes:

1. Job/Process Tables 2. File System Tables 3. I/O Tables

We'll describe the major tables in these classifications and explain the appropriate inter-relationships. Following the table explanations, we'll present examples to aid understanding.

JOB/PROCESS TABLES

THE DATA STACK -- Every process has a stack associated with it. We've all seen the familiar pictures of the stack showing the area from DL to Z. What these pictures don't show is the area beneath DL. This area is called the PCBX and is further divided into three areas:

PXGLOB - area containing JOB TABLE pointers & indices

PXFIXED - misc. information about this process

PXFILE - the file system section of the PCBX

Since a data stack exists for all processes in a job, the PXGLOB area is almost identical in all stacks belonging to the same job. When a process terminates, its data stack goes away. A job or session is just a special type of process. Therefore, if you wish to make any changes to the PCBX area, you must do it on the stack of your main process in order for it to stay until you log off. Since the job-related information in the PXGLOB area is the same for all processes in a job, you can use the PXGLOB area of the process (program) you are running to get to your job tables.

The PXGLOB area has pointers to the following job tables: JMAT, JPCNT, JDT, JIT, and JCUT. The following is a brief description of these tables. Refer to figure 1 for a related diagram.

JMAT -- The JMAT (Job Master Table) is one table presently located at DST %31. This table contains a header entry followed by one entry per logged on job or session. An entry in the JMAT contains (among other things): the job/session number, user, account, group, and job name, input and output device numbers, time and date of logon, PIN of the job/session main process, log on priority, CPU time limit, etc.

JPCNT -- This table (Job Process Count Table) contains a byte for each logged on job or session and is used in the allocation and deallocation of SIRs for jobs.

JDT -- A JDT (Job Directory Table) exists for each job on the system. This is unlike the JMAT and JPCNT which have all jobs and sessions contained in a single table. The JDT is composed of five smaller tables. These five tables contain information regarding data segments, temp files, file equations, line equations, and JCWs used by the job.

JIT -- The JIT (Job Information Table) is similar to the JDT in that a different data segment is used for each job on the system. A job's JIT contains information such as the job's account security, group security, account name, home group, logon group, user name, job name, local attributes, ALLOWed commands, account- ing information, directory pointers for the account and group, the PIN of the main process, etc.

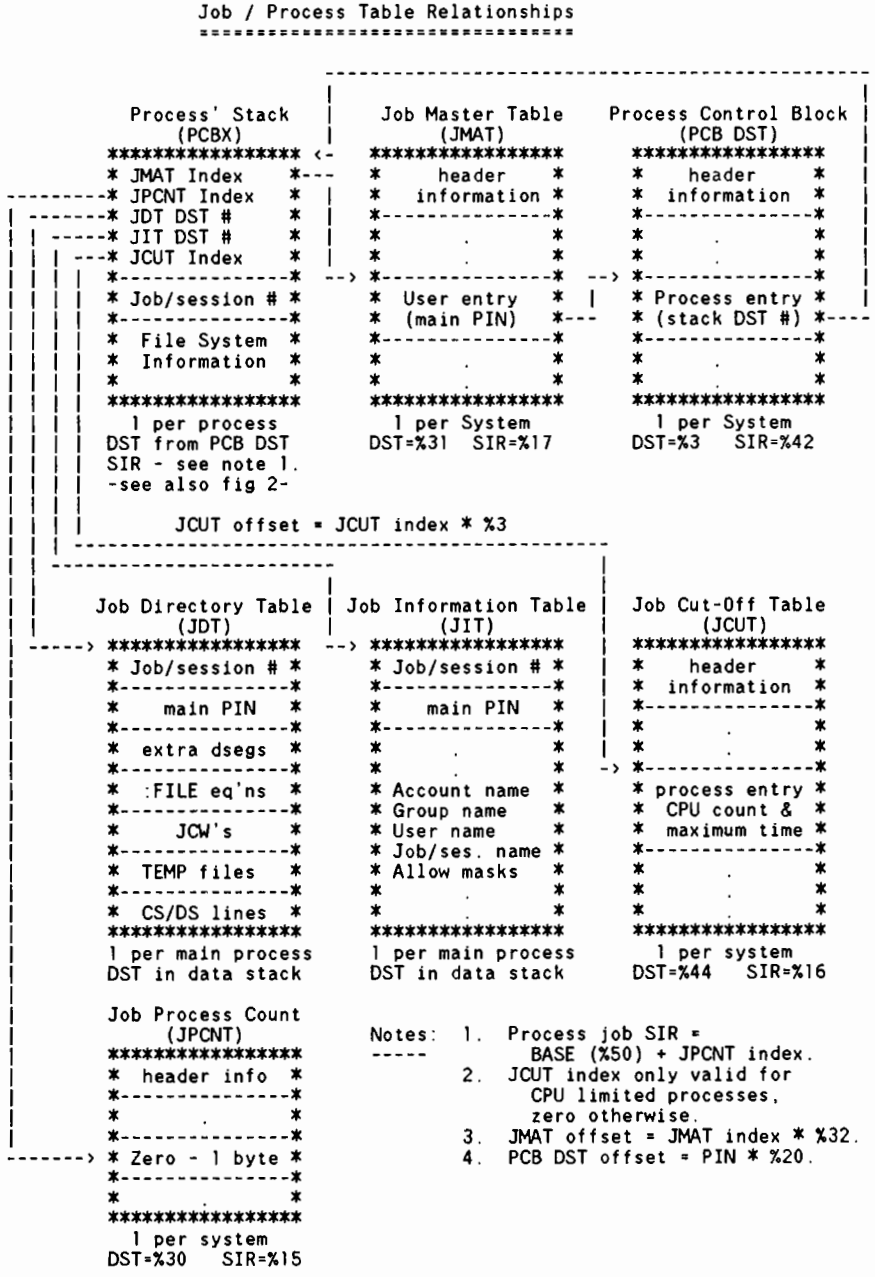
JCUT -- The JCUT (Job Cutoff Table) is a single table located at DST %44. This table contains one entry for each CPU limited job or session. If your job or session has a limited number of CPU seconds, it will have an entry in this table. When your CPU limit is reached, you will be logged off.

The interesting job tables, in our opinion, are the JMAT, JDT, and the JIT. Of these three, only the JMAT has the format of having one entry per job instead of one table per job. Since the JMAT has the job/session numbers in it, it is the prime starting place for any program looking for a particular session, or job; or a program which will print something for all logged on jobs and sessions. The MPE command :SHOWJOB simply scans the JMAT and reports it in a meaningful format.

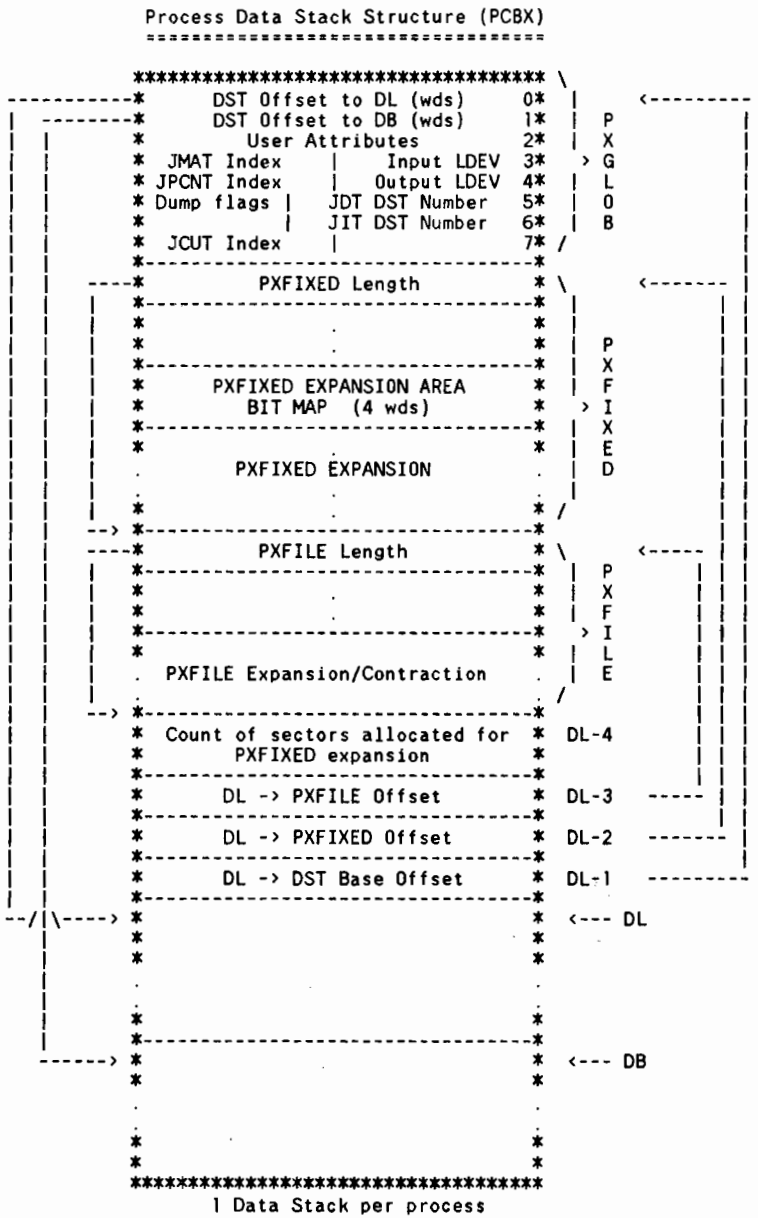
An interesting exercise for the novice PM programmer is to write a program to print a SHOWJOB by reading the JMAT. SHOWJOB tells you the day of the week that a person logged on, but did OPERATOR.SYS log on last week or last month? It's in there, but MPE won't tell you.

As mentioned earlier, if you want to change anything in the PCBX area, you've got to do it on the stack of your main process. To do this, you need to know the DST number of your main process' stack. This number is stored in the PCB (Process Control Block) DST (data segment #3) with one entry for each process -- PROCESS not JOB! Remember, a job or session is a special type of process. The PCB entries are indexed by the PIN (Process Identification Number) of the process whose entry you are looking for. The PIN of your job/session main process can be found in your JMAT entry, your JDT, and your JIT. Take your pick. Figure P1 shows a listing of a procedure which will return your main process' PIN and the DST number of your main process' stack. This procedure GET'MAIN calls a procedure GET'DATA which uses the MFDS (Move From Data Segment) machine instruction to retrieve data from any data segment on the system. The comments in the two procedures make them virtually self-explanatory. Figure 1 shows the relationships between the job/process tables. For specifics, refer to the MPE System Tables manual.

Job / Process Table Relationships

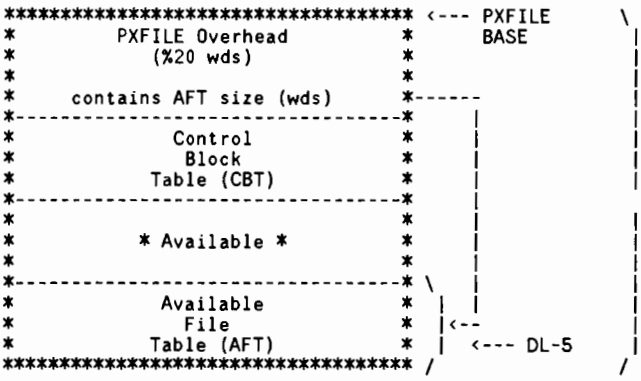


-- Figure #1 --

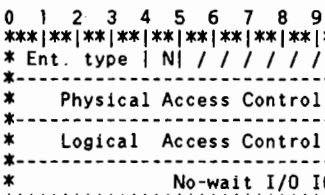

Process Data Stack Structure (PCBX)
DST Offset to DL (wds) 0*
DST Offset to DB (wds) 1*
User Attributes 2*
JMAT Index | Input LDEV 3*
JPCNT Index | Output LDEV 4*
Dump flags | JDT DST Number 5*
| JIT DST Number 6*
JCUT Index | 7*
PXFIXED Length
PXFIXED EXPANSION AREA
BIT MAP (4 wds)
PXFIXED EXPANSION
PXFILE Length
PXFILE Expansion/Contraction
Count of sectors allocated for PXFIXED expansion
DL -> PXFILE Offset
DL -> PXFIXED Offset
DL -> DST Base Offset
DL-4
DL-3
DL-2
DL-1
<--- DL
<--- DB
PXFIXED
PXFILE
1 Data Stack per process

-- Figure #2 --

File System Area of Process Data Stack (PXFILE)



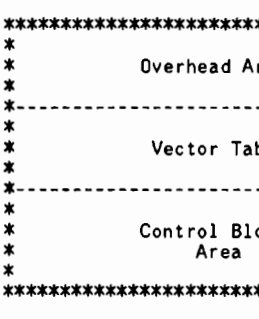
The Available File Table (AFT) is indexed in reverse order by file number, i.e. the entry for file #1 is at DL-8 thru DL-5, file #2 is at DL-12 thru DL-9, etc. For unused file numbers, the entry is all zeros. The number of entries in the AFT (# of files) = AFT Size from PXFILE Overhead / 4. There is a 4 word AFT entry for each file of the process, formatted as follows:



Entry Type: 0-File system
1-Remote file
2,3-DS files
4,5-CS files
6-KSAM files
8-MSG files

N: File is $NULL, no ACB vectors used.
LACB: Only for multi-access files.
PACB: Valid for all files except $NULL.

The PACB and LACB vectors, are indexes into tables known as Control Block Tables (CBT). The general structure of a CBT is:



The ACB vectors of the AFT entry consist of a six-bit vector table entry number & a Data Segment Table number containing the CBT. *All* pointer information for the CBT's are relative to the CBT beginning address. The vector table entry contains a pointer into the Control Block Area which is then the actual LACB, PACB or FCB. The CBT may be located in two places: 1) on a process' data stack, or 2) in a separate segment. Since all pointer addressing is relative to the CBT starting address, care must be taken if the CBT is located on a process' stack. (You can determine if a data segment is a process' stack by checking flags contained in the Data Segment Table (segment #2) entry for that segment). The same repetitive procedure is used when finding the LACB, PACB and FCB.

I/O TABLES

The input/output tables in MPE have a much more complex structure than the job tables. We don't recommend modification of these in any program. The important I/O Tables in our opinion are the LPDT -- Logical/Physical Device Table, DIT -- Device Information Table, ILT -- Interrupt Linkage Table, and the IOQ -- Input Output Queue. We use these tables solely for verification in our programs, to make sure we're communicating with the device we think we are. It's very possible to attempt to read or write beyond the physical limits of a disc if you don't know the type of disc you're using. Also, you wouldn't want to think you're writing to a terminal and actually be writing to a disc. For these reasons, we check device information in the I/O Tables. The System Tables Manual contains an entire section on I/O tables for the interested reader.

All I/O requests are processed through the IOQ Table. A procedure called ATTACHIO handles the interfacing to the IOQ. All our information on ATTACHIO has come from what we have read in contributed programs, so we know what works but not exactly how or why. A page toward the end of this paper explains parameters to ATTACHIO as we know them. Use this procedure with extreme caution.

FINDING OUT MORE ABOUT MPE

So you haven't had enough? You want to find out more? Here's how we continuously learn more about MPE internals.

1. Get yourself an up to date set of manuals. We specifically recommend the following:

A. HP3000 System Reference Manual (HP Part # 30000-90020) This manual is found at most HP3000 sites, but few people bother to read it. It is an excellent source of information on stack operation, code segments, data segments, and the I/O system. Skim the book and make notes (mental or otherwise) on what information is in the book and where to find it in case you need it.

B. HP3000 Machine Instruction Set (HP Part # 30000-90022) Another seldom-referenced manual found in most HP3000 shops, it is often hidden in the same binder as the Sysyem Reference Manual. We recommend that you become familiar with HP3000 machine language for reasons mentioned later. Perhaps the most valuable page in this manual is page A-1. Always look there first -- it's a real time saver.

C. HP3000 Intrinsics Manual (HP Part # 30000-90010) For anyone programming in SPL, this is a MUST. Pay particular attention to what the intrinsics DON'T do and you've got a nice privileged mode program to write.

D. Programming Language reference manual of your choice. As mentioned earlier, we recommend SPL.

E. MPE System Tables Manual -- Another MUST for the serious privileged mode programmer. We hang on to our old System Tables Manuals even though they are now outdated and often totally inaccurate. It's interesting to see the advances made in MPE over the years. Looking back also gives us a good idea as to what types of things are likely to change from one version of MPE to the next.

2. The second thing you can do to find out more about MPE is to learn to read HP3000 machine language. You don't need to have every mnemonic mnemorized, but you should get intimate with the Machine Instruction Set manual. The next best thing to knowing something is knowing where to find it. You'll be surprised how fast it comes to you.

3. Thirdly, use your newly acquired knowledge of machine language to set breakpoints in HP supplied code. How does SPOOK open a spool file? Set breakpoints, use DEBUG, step through the code and find out. It's easier than you think.

4. Equally enlightening is the use of so-called "program dumpers". SEGMENTER can tell you a lot about SL.PUB.SYS. Use it to find the location of "interesting" procedures. Numerous contributed programs will tell you what external procedures are used by a program. So, you find out that SPOOK calls a procedure called FSOPEN. Find it in SL.PUB.SYS using SEGMENTER then "decompile" it using DEBUG with the C (CODE) option or the

latest version of DECOMP which lets you look at SL procedures.

5. Look in the contributed library for PM programs. We try to check out all new ones on each tape we get. One rule of thumb: No source -- it gets purged. It's surprising how many HP employees with a knowledge of MPE internals and PM techniques will contribute programs (often anonymously). You can frequently get good ideas or find out how to call those "undocumented uncallables". You can always customize the program to meet your specific needs (now that you know how). One final word of caution: don't assume that all privileged mode contributions are clean. For that matter, don't assume that ANY PM contributions are clean. Treat each one with caution, and evaluate the source thoroughly before you even THINK of typing :RUN.

6. Finally, you've got to write privileged mode programs. You can't learn anything unless you do it. Be prepared for the worst and don't be afraid to try things. That's easy for us to say, we're not in charge of keeping your computer running.

Remember that HP doesn't like for you to use privileged mode (and we don't blame them). When you get system failures, you should remove PM capability from any account having it (except SYS) so you can be ABSOLUTELY sure that your PM program is not the culprit.

HOW TO MAKE YOUR SPL PROGRAM RUN IN PRIVILEGED MODE

If you are writing a program:

No special capabilities are needed to compile it. The USER (and thus the ACCOUNT) needs PM capability in order to PREP with CAP=PM

$CONTROL PRIVILEGED in your source causes a program to start in privileged mode when it is run, unless ;NOPRIV is specified on the :RUN command.

Without $CONTROL PRIVILEGED, you can still call the GETPRIVMODE intrinsic at any time, if and only if your program has been :PREPped with CAP=PM

SAVEd programs must reside on a group with PM capability.

If you're running $OLDPASS or a TEMP file, then the user must have PM capability. This prevents you from naming a temp file X.PUB.SYS and running it.

If you are writing a procedure:

Privileged procedures must reside in an SL on a privileged group.

The USER (and thus the account) needs PM capability to be able to -ADDSL a segment containing a privileged procedure in SEGMENTER

Using OPTION PRIVILEGED, you are running in privileged mode as soon as you call the procedure.

Using $CONTROL PRIVILEGED, you can use GETPRIVMODE to jump into privileged mode when you desire.

OPTION PRIVILEGED puts you into privileged mode whenever you call ANY procedure in the same segment as the procedure declared OPTION PRIVILEGED.

EXAMPLES USING PM DEBUG

As mentioned earlier, PM DEBUG is an excellent way to make small changes to a system table or to take a quick look at one. Here are some examples to illustrate.

To enter privileged DEBUG, make sure you are logged on to a user having PM capability. Then type the MPE command :DEBUG. DEBUG is not very user-friendly and it responds with a ? prompt.

```
:DEBUG

*DEBUG* PRIV.A30.14
?
```

To look at a system table, use the DDA (Display Data Segment) command. The DDA is followed by the data segment number you wish to examine, an optional offset into the data segment, optional number of words to display, and optional display mode. Everything entered into DEBUG and displayed by DEBUG is in octal unless otherwise specified. So, suppose we want to look at the number of seconds allowed for a user to log on. The System Tables Manual tells us that this number is at word %120 in the SYSGLOB (System Global Area). It also tells us that the SYSGLOB is DST #5. So, to display this value in decimal (base 10), you would type:

?DDA5+120,I

and DEBUG will print something like:

DA5+120 +00120

which tells us that all users have 120 seconds to log on after they establish connection with the computer. Shortening this time to a ridiculously low value is one way of keeping people off the system. Similarly, lengthening the amount of time may be desired if you have many new users who have a great deal of trouble typing their :HELLO command. You can change this value on the fly by typing:

MDA5+120,I

Which does the same thing as the DDA command, except that it prompts you for a new value. Remember that your input is octal unless you specify otherwise. To enter a decimal number, prefix your entry with a # sign. For example, to change the log on time to 4 minutes instead of two minutes:

```
You type:        ?MDA5+120,I
DEBUG says:      DA5+120    +00120   :=
```

At the := prompt, enter your new value. Four minutes is 240 seconds. Prefix it with a # symbol like this:

DA5+120 +00120 := #240

and it's changed. Presto. No cold start. You can use this technique to change many things without having to take down the system for a cold start. If you need to change the maximum extra data segment size or the maximum number of extra data segments per process, for example, these numbers are stored at locations DA5+111 and DA5+112 respectively.

For example, increase the maximum number of extra data segments per process to 32.

```
:DEBUG
*DEBUG* A30.14
?DDA5+112,I

DA5+112     +00004

?MDA5+112,I

DA5+112     +00004   :=#32

?R
```

Note that the R command is used to exit DEBUG. In this example, the previous maximum number was 4. This change takes effect immediately; there is no need for a cold load.

Suppose you want to check on a job or session to find out what command it is currently executing (i.e. How far along is your job?) Try the following:

1. Locate the main pin of the desired job or session using the MPE :SHOWQ command. The PIN will be preceded by the letter M and followed by the session number. For example:

C M14 #S639

tells you that session #639's main PIN is 14 (decimal).

2. Enter PM DEBUG and do the following steps. Find the size of a PCB entry. This is located at location 1 in the PCB DST, which is DST #3. The number on most recent versions of MPE is 16 (%20) but check to make sure.

```
:DEBUG
*DEBUG* PRIV.A30.14

?DDA3+1,I
DA3+1    +00016
```

3. Calculate the offset into the PCB DST for this process by multiplying the PIN times the entry size. In this case, our offset is equal to 16*24 which equals 224 (decimal). This number is the starting location of the PCB entry for the process' main PIN. We want to examine word #3 of the

process' PCB to get the DST number of its stack. We can do this manually, or let DEBUG handle the arithmetic as follows:

```
                    ?DDA3+#224+3
OR WE COULD ENTER   ?DDA3+#16*#14+3

                    DA3+343     005020
```

4. Manually extract bits 1:10 from this value to get the stack DST number. In this example, %005020.(1:10) = %120

5. Word number 1 in the stack DST gives us the offset to the stack's DB area.

```
?DDA120+1
DA120+1    000444
```

6. We want to look at the string located at DB+1, so by adding one to the DB offset, and dumping the area in ASCII, we can look at the last command executed by this main process.

```
?DDA120+444+1,30,A
```

The command will be displayed, and it may be incomplete or followed by garbage. We have no way of determining the length of the command except by inspection. The command ends with a carriage return, which is displayed in the ASCII dump as a period.

Granted, this procedure is somewhat complex, and you'd probably be better off writing a program to handle such a task. The example is only intended to show how easily information can be retrieved from system tables.

We have written other programs which do nice things in Privileged Mode. It's not hard to switch to another group without logging on again. This involves verifying that the group exists, looking up its pointers in the directory, and putting these pointers in the right places in the job tables. The name of the group you are logged onto must be changed in the job tables also, or the SHOWME command will be inaccurate. What the contributed programs don't do (for the most part) is update the directory with pertinent information. It may or may not be important to you if your GROUP CPU seconds and CONNECT time values are accurate. We think they should be updated when you switch to a new group. Another item in the directory that needs to be updated and is often ignored is the usage counter. Each account and group within the account has a counter that keeps track of the number of users currently logged on to the account. This prevents someone from purging a group or account while it is in use, leaving the user of it in limbo. With most contributed group and account jumping programs, these group usage counters get out of whack.

Another nice program is one which will allow you to do a "GOTO" while executing a UDC. MPE provides an IF condition, but no GOTO. The most popular application of this is when you have a MENU type environment. Your UDC runs a program (or any number of programs) then when a program ends, you want to start the entire command all over again. To exit the loop, set a JCW in your menu program (perhaps triggered by a function key) and test it using the MPE :IF command. Our program uses the PARM value as a relative index for branching within the UDC. For example, when run with PARM=-5, the program moves the pointer back 5 lines in the UDC. Similarly, PARM=3 skips execution of the next three lines in the UDC. This has proven to be a very powerful enhancement to UDCs.

The simple program operates as follows. It uses our procedure GET'MAIN to locate the DST number of the main process' stack. On the stack, DB+%1635 contains the record number in the UDC file of the next line in the UDC to be examined for execution. The program simply adjusts this value up or down depending on the value of the PARM. Note that because recursion is used when executing UDCs, this will work only on your outermost UDC. For example if your MENU command invokes another UDC, such as RUNP, don't expect the UDCLOOP program to work correctly in the RUNP command.

CONCLUSION

Privileged Mode is not for everybody. We have attempted to explain it here in detail, and to cover a few system tables to get the interested programmer started. We may have tried to cover too much material in this paper, but we want to provide as much information as possible to assist any present or future privileged program writer. We invite questions and/or comments, and we apologize in advance since we may not be able to reply to all of them. Thank you.

Here are some other userful (and sometimes necessary) MPE internal procedures, that we have come across in contributed software programs. These procedures are all found in the system SL, SL.PUB.SYS.

```
  I                IV
FLAG:= GETSIR (SIR'NUMBER);
```

This procedure is used to lock a SIR (system internal resource) to prevent data accessing problems when more than 1 user may be involved. The SIR must be 'unlocked' before termination of the program or a system failure will result. The value of 'FLAG' is passed to the procedure RELSIR.

```
             IV       IV
RELSIR (SIR'NUMBER, FLAG);
```

This procedure is used to unlock a SIR after it has been locked by GETSIR. FLAG is the value returned by the corresponding GETSIR. A locked SIR must be unlocked (released) before ending the program, or a system failure will result.

```
  L
FLAG:= SETCRITICAL;
```

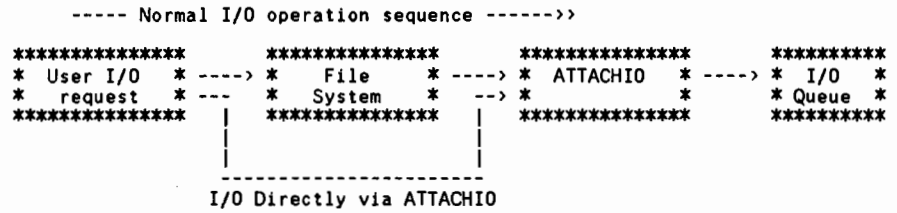
This procedure causes the system to only execute this process until RESETCRITICAL is called. If SETCRITICAL has been called, a RESETCRITICAL must be executed before terminating the program or a system failure will result.

```
                IV
RESETCRITICAL (FLAG);
```

This procedure reverses the critical status of a process after a SETCRITICAL. This must be called if a process has been made critical or a system failure will result. FLAG should be zero. Use of SETCRITICAL & RESETCRITICAL will insure (like the use of SIR's) that no other process will be accessing data you are attempting to modify.

ATTACHIO

Normally, all Input & Output is accomplished via calls to file system intrinsics. However, through the use of privileged mode, all the normal checks and limitations imposed upon the user by MPE may be by-passed. This is achieved by calling an MPE internal procedure called ATTACHIO. ATTACHIO simply formats parameters and queues requests to the device drivers.



The parameters to ATTACHIO are as follows:

```
            IV     IV    IV    IV    IV     IV    IV     IV     IV
  ATTACHIO (LDEV, QMISC, DSTX, ADDR, FUNC, COUNT, PARM1, PARM2, FLAGS)
```

Parameters

LDEV	Logical Device Number for I/O request.
QMISC	Request state & flags. Only observed to be zero.
DSTX	DST of target data area. Only observed to be zero.
ADDR	Address of data buffer to be written or read into.
FUNC	Function Specification. 0 - Read from specified LDEV. 1 - Write from data buffer to LDEV.

COUNT Transfer count. Number of words (or bytes if COUNT < 0) to be read or written.

PARM1 & 2 Zero for terminal I/O. Contains disc address for disc I/O.

FLAGS Observed to be:
1 - Disc I/O.
%401 - Terminal I/O.

ATTACHIO returns two words (double integer) after execution. The first word contains the actual transfer count, exactly like the count returned by FREAD, in the second word, bits 8:8, the general status is given:

0 - Awaiting completion.
1 - Successfully completed.
2 - EOF detected.
3 - Unusual Condition
4 - Irrecoverable error.

This procedure is probably the most powerful (and consequently most dangerous) procedure available to the privileged mode user. Use of this procedure should never be done casually, and every possible check should be performed. (Such as verifying LDEV information from internal device tables, ADDRESS information from device type & subtype tables, etc.).

```
LOGICAL PROCEDURE GET'DATA (DST'NUM, OFFSET, COUNT, ADDR);
VALUE DST'NUM, OFFSET, COUNT;
LOGICAL DST'NUM, OFFSET, COUNT;
LOGICAL ARRAY ADDR;
OPTION CHECK 3;

BEGIN

ENTRY PUT'DATA;

INTRINSIC GETPRIVMODE, GETUSERMODE;

LOGICAL ARRAY ENTRY'BUF (0:3);
LOGICAL WRITE'DATA;

WRITE'DATA:= FALSE;
GO TO LET'S'START;

PUT'DATA:       << ENTRY FOR WRITING TO DATA SEGS >>
WRITE'DATA:= TRUE;

LET'S'START:

GETUSERMODE;    << JUST TO BE SAFE >>

ADDR (0):= ADDR (0);    << MAKE SURE IT WILL FIT >>
ADDR (COUNT - 1):= ADDR (COUNT - 1);

GETPRIVMODE;    << LET'S SEE IF THERE IS SUCH A SEGMENT >>

TOS:= @ENTRY'BUF;
TOS:= 2;
TOS:= 0;
TOS:= 4;
ASSEMBLE (MFDS 4);  << GET DST ENTRY 0 >>

IF DST'NUM > ENTRY'BUF (0) THEN GO RETURN'FALSE; << BAD DST >>

TOS:= @ENTRY'BUF;
TOS:= 2;
TOS:= DST'NUM * 4;
TOS:= 4;
ASSEMBLE (MFDS 4);  << GET DST'S ENTRY >>
```

```
IF ENTRY'BUF (0).(3:13) = 0 THEN GO RETURN'FALSE;  << NOT USED >>
ENTRY'BUF (0):= ENTRY'BUF (0).(3:13) * 4;   << LENGTH OF SEGMENT >>

IF OFFSET + COUNT > ENTRY'BUF (0) THEN GO RETURN'FALSE; << BAD LEN >>

<< LOOKS OK, GET THE DATA >>

IF WRITE'DATA THEN BEGIN
  TOS:= DST'NUM;
  TOS:= OFFSET;
  TOS:= @ADDR;
  TOS:= COUNT;
  ASSEMBLE (MTDS 4);    END;   << WRITE THE DATA >>


ELSE BEGIN
  TOS:= @ADDR;
  TOS:= DST'NUM;
  TOS:= OFFSET;
  TOS:= COUNT;
  ASSEMBLE (MFDS 4); END      << GET THE REQUESTED DATA >>

GET'DATA:= TRUE;
GETUSERMODE;
RETURN;

RETURN'FALSE:

GET'DATA:= FALSE;
GETUSERMODE;
END;

LOGICAL PROCEDURE GET'MAIN (MAIN'PIN, MAIN'STACK'DST);

LOGICAL MAIN'PIN, MAIN'STACK'DST;
OPTION CHECK 3;

BEGIN

INTRINSIC GETUSERMODE, GETPRIVMODE;

LOGICAL POINTER PXGLOB, S'0=S-0;
LOGICAL JIT'DST, PCB'DST, PCB'ENTRY'SIZE, PCBPT;

PCB'DST:= 3;

GETPRIVMODE;

PUSH(DL);            << PUT CONTENTS OF DL ON STACK >>
TOS:= S'0(-1);       << PUT ON CONTENTS OF DL-1 >>
ASSEMBLE (SUB);      << THE DIFFERENCE IS PXGLOB BASE >>
@PXGLOB:= TOS;

JIT'DST:= PXGLOB(6).(6:10);  << GET THE JIT DST NUMBER >>

GETUSERMODE;

IF NOT GET'DATA(JIT'DST, 10, 1, MAIN'PIN)
     THEN GO RETURN'FALSE;
MAIN'PIN:= MAIN'PIN.(8:8);

IF NOT GET'DATA(PCB'DST, 1, 1, PCB'ENTRY'SIZE)
     THEN GO RETURN'FALSE;
PCBPT:= MAIN'PIN * PCB'ENTRY'SIZE;

IF NOT GET'DATA(PCB'DST, PCBPT+3, 1, MAIN'STACK'DST)
     THEN GO RETURN'FALSE;
MAIN'STACK'DST:= MAIN'STACK'DST.(1:10);
```

```
GET'MAIN := TRUE;
RETURN;

RETURN'FALSE:
GET'MAIN := FALSE;
END;
```

Joseph C. Felix is a Software Design Specialist for Educational Computer Systems, Inc. of Cincinnati, Ohio. He has a B.S. from the University of Cincinnati in Information Processing Systems. As one of the founders of Educational Computer Systems, Joe has been with the company since its inception in 1977. He has been writing SPL programs using privileged mode since 1978. His work with HP operating systems dates back to 1974 when he customized TSB, the HP2000 operating system, while employed by Cincinnati Public Schools. In his free time, Joe enjoys amateur radio, community theater, music and electronic tinkering.

Chris Hauck is a Software Design Specialist for Educational Computer Systems, Inc. of Cincinnati, Ohio. He has a B.S. degree in Computer Science from the University of Dayton. He has been using HP computer systems since 1976, beginning with the HP 2000. In 1979, at Educational Computer Systems, he began using SPL and privileged mode on the 3000 and has since written countless privileged mode programs and procedures.

---

ingenieurbüro jörg grössler gmbh
- edv - beratung & system - unterstützung -

rheinstraße 24
1000 berlin 41
030/852 70 27

Dokumentation

1.84

AN ALTERNATE TAPE BACKUP SYSTEM REPLACING STORE AND SYSDUMP

AUTHOR: DIPL.-ING. JÖRG GRÖSSLER

DATE: 11/15/83

An Alternate Tape Backup System Replacing STORE and SYSDUMP

1. Introduction

Since 1975, I am using the HP3000 Computer System. Everybody knows that the only available system to backup disc files on magnetic tapes is STORE and SYSDUMP which both have the same internal structure. Very soon after I had started using the system the idea for an alternate backup system was born which should eliminate most of the disadvantages of STORE and SYSDUMP.

Now, in 1984, these plans have come to a point where we can start implementing a new tape backup program. This paper will present the basic philosophy of the new tape backup system and outline the major differences to the existing STORE and SYSDUMP.

2. How STORE and SYSDUMP work

First of all it has to be said that STORE and SYSDUMP produce tapes with compatible format. The difference between SYSDUMP and STORE tapes is that SYSDUMP uses the first two files of the tape which are empty on ordenary STORE tapes. In the first file, SYSDUMP puts in permanent table information (for example the RIN table) and the system directory. In the second file system programs, the SL and non standard driver are stored.

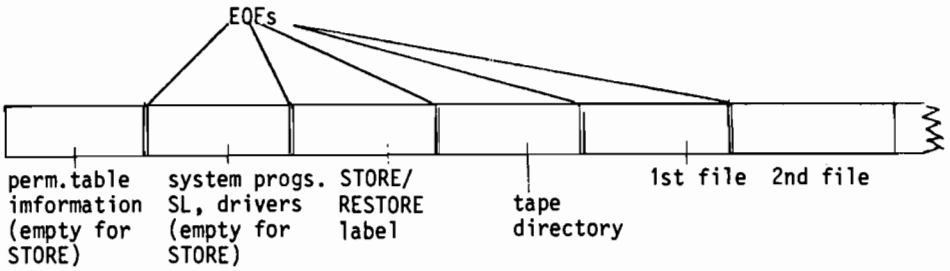


Fig. 1: general STORE/SYSDUMP format

Figure 1 shows the general STORE/SYSDUMP format for the first volume of a STORE/SYSDUMP tape. The so called STORE/RESTORE label contains an identification which enables RESTORE to identify a STORE tape. The different formats (HP,for example, switched from 1 K to 4 K blocks in the late '70s) are also mentioned here.

The contents of the tape directory is very easy to explain. It just contains the fully qualified file names of each file on the tape (file name, group name, account name). No further information is stored.

For each file on the disc there is one file on the tape. It contains the information sectorwise bound together in 4 K blocks. Each file starts with the MPE file label (one sector of information) possibly followed by user labels and the actual contents of the file. In case the file contains less than 4 K words of information the block stored on the STORE tape is also reduced accordingly.

Figure 1 shows the format of STORE/SYSDUMP tapes for the first volume. Each additional volume contains as the first file an additional STORE/RESTORE label and the volume directory in one file. The volume directory is similar to the tape directory except it only contains those file names which are stored on this tape and the rest of the tapes since STORE cannot avaluate how many files will fit on a specific tape reel.

The format of STORE tapes for labeled tapes is very similar for the first reel except that additional files are produced by the tape label subsystem. The format of each additional volume is different due to the fact that the tape label subsystem handles multiple reels itself. The volume directory for subsequent volumes, therefore, is not stored as the first file of each additional volume. This fact is not very important to understand how the alternate backup system works.

3. The Tape Format of the New Tape Backup System

The new tape backup system uses a very enhanced tape format. Much more information is stored in the tape directory and we will see later on how this effects the handling of this new system.

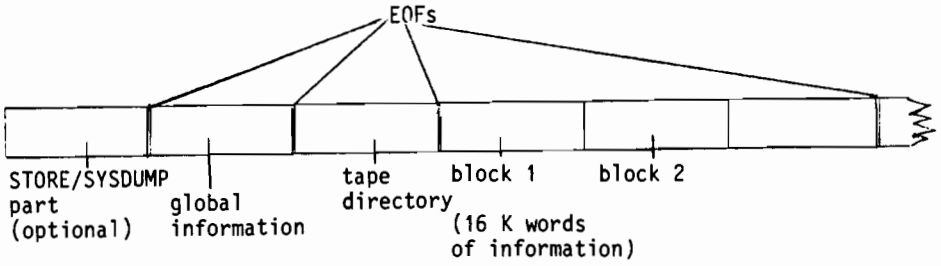


Fig. 2: format of new backup tape

First of all you will see that there can be any STORE/SYSDUMP prior to the actual new backup tape. This is done for two reasons:

- On an ordinary new backup tape the program to restore the tape will always be stored in ordinary STORE format at the beginning of the tape. This ensures compatibility with systems where the new backup system is not installed.

- This format also makes it possible to produce a complete backup on one single set of tapes. In this case the STORE/SYSDUMP part contains an ordinary SYSDUMP including important files from PUB.SYS and the operating system.

There is not much to say about the part which contains the global information. Its structure will be similar to the STORE/RESTORE label.

The tape directory will contain much more information about the files stored on the tape. For each file there will be a 64 words entry which contains most of the information out of the file label.

The actual contents of the files are stored in continuous blocks. Each of these blocks contains 16 K words of information. There is one EOF after every 10th block.

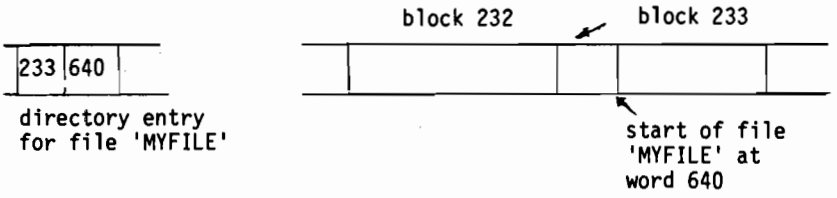


Fig. 3: identification of files

Figure 3 shows how files are identified in the new backup tape. Positioning a file for a RESTORE can be done very easily. Knowing the block number, the system will perform 'BLOCKNUMBER/10' number of SKIPEOF. Then it will skip 'BLOCKNUMBER MOD 10' blocks to reach the actual start point.

4. Results

The introduction of a new format for a tape backup system makes it possible to provide better features than the existing STORE system.

- There is less tape consumption because the blocks of information are larger and small files are packed into the same block and not split into even smaller blocks with one EOF at the end of each file. This measure saves between 10% and 20% of the tape needed for a backup.

- Compressing the data (stripping off zeros and blanks) gives an other 40% to 50% of less tape quantity.

- A considerable improvement of the restore speed can be expected because of the new layout of the tape directory. Since all the information of the file label is stored in the tape directory, a parallel program can go ahead and start building files before the actual location of the file is reached. In the existing system, RESTORE is only able to build files when the file label which is part of the file itself, is read from the tape. This means an average delay of approx.1 sec. for each file on the tape, regardless of the speed of the tape drive.

- The new tape directory layout also makes it possible to append and replace files to/on an existing tape. At the moment it is necessary to restore the entire tape and store it again when you want to append one single file which is a very time consuming procedure.

- The new backup tape will have its own label identification which means that the standard tape label subsystem will not be used. This makes sense since the tape label subsystem produces a lot of overhead and inhibites the system to use the faster ATTACHIO instead of the file system, and since the tape label subsystem can never be used with SYSDUMP tapes.

- The additional information in the tape directory will also make it possible to decide whether or not a RESTORE can be performed without problems prior to the point where the actual file is on the tape. This means that the user will have the information that there are files on the tape with a creator which is not stored in the system directory at the beginning. So far, this information came out at the very end of the RESTORE and in most cases the entire procedure had to be re-done. With the new system, the user can decide at the beginning whether or not he wants to proceed or may put the creator into the system directory.

All these new features make it worth to rebuild the entire layout of a backup tape. The disadvantage of not being compatible with the existing STORE/RESTORE system is made less severe because on each tape there will be at least one file in the existing STORE format, and this is the program to restore the tape.

I am very optimistic that this new backup system, once it has been introduced, will solve many of the existing problems to backup discs on tapes, and that this is something the users have already waited for a long time.

DISASTER PLANNING AND RECOVERY

by Dennis Heidner
Boeing Aerospace Company

ABSTRACT

This paper covers disaster planning and recovery. The author discusses traditional disaster planning methods and presents the idea of disaster levels. The author develops guidelines which can be used to perform risk analysis and write a comprehensive disaster and recovery plan. Finally the author discusses applicable tools which are available from the contributed library and third party vendors and ways these tools can aid your planning and software preparation.

INTRODUCTION TO DISASTER PLANNING AND RECOVERY

When the subject of a disaster plan is brought up, very often your co-workers react like you are crying wolf. "Why", they ask, "should we worry about the computer system when it has been running so well. Do not tempt fate by preparing for the worst!" Unfor- tunately, this is the old "head in the sand" idea. Any sane and intelligent data processing manager MUST pursue and develop a contingency plan for the loss of the computing facility; such plans are called DISASTER PLANS.

What is a disaster? Well, our friend Noah Webster says that a disaster is 'any happening that causes great harm or damage; serious or sudden misfortune; calamity'. The complete destruction of the computer facility could be no doubt called a sudden misfortune or calamity. By the same token could we not also call the temporary loss of the computer, at the end of the month when it is printing paychecks, a calamity?

When the word "disaster" is mentioned, almost always the first thought is that of an 'act of God' such as a flood, earthquake, volcano, tornado, hurricane, fire, etc. However disasters can also be the result of deliberate acts of vandalism, security breaches, critical application software failures, or even catastrophic hardware or software failure on the part of the computer or MPE! In addition the one element that we almost always forget is human error. Consider this situation: you have begun a WARMSTART after shutting the computer down so you can change the date and time. The portion of MPE called INIT has just started and for some reason the operator realizes that the wrong tape reels are on the drive so (s)he halts the computer. Unknowingly your operator has interrupted MPE at a critical point because the computer has already changed information in the volume labels. Now when the next WARM or COOLSTART is attempted, the computer will fail to boot!

DISASTER LEVELS: NOT ALL DISASTERS ARE CREATED EQUAL

How calamitous a disaster is depends entirely on the potential (or actual) loss that could be (is) incurred. It is possible to have a 'disaster' on a new computer installation before it is in production use such that the amount of loss is very insignificant.

What we want is some type of plan that will handle both the minor mishaps and the major disasters. The plan should be easy to follow, well thought out, and practiced! How do we develop such a plan?

It is here that I differ from the traditional concept of disaster planning. In the traditional method, plans are made only for the worst case loss. But remember the possibility of the case' occurring is actually quite small, wh it is very likely during the life of your ma

that you will experience a 'minor mishap'. The first step is to determine the potential damage if you were to lose the computing facility for a short interval, a somewhat longer interval, and finally completely. Once you know the potential losses you can try to develop a plan that pays off for each case. Besides any 'major disaster plan' must draw on the techniques used to recover the application after minor mishaps. Why not, then, develop a plan which covers more then one level of disasters?

Designing a disaster plan for several crucial levels has distinct advantages: First, the plan for a major disaster must draw on the techniques used at all levels. Next it is a convenient way to document the stages at which you escalate the involvement of key personnel. In addition the marginal costs in the disaster plan are much more apparent. This allows better informed management decisions for the risk analysis and protection costs.

In many cases the multi-level disaster plan is simply a formally documented version of the currently used undocumented procedures that the data processing staff has already been using. At our site the disaster plan has three levels:

Level 1 The computer is down for up to four hours following an interruption. The time is spent recovering and verifying that the system/application is intact. During this time transactions are handled with paper and pencil. The work is then caught up over the next several days.

Level 2 The computer will be down for more than four hours but less than twenty-four. As part of the recovery procedure a special team is added to work a second shift entering the transactions that were collected on paper during the first shift. The following day exceptions are corrected by those who hand-wrote data the day before.

Level 3 The computer will be down for more than twenty-four hours. During this time transactions are made by hand. The special data entry recovery crew is standing by. The site team is actively considering the alternate site. If one is made available then the application is moved to the alternate site. Once installed the data base is checked for damages, and corrected. After the intial check out has been completed then the special data entry team is used to enter the accumulated paper work.

RISK ASSESSMENT

Risk assessment involves determining the cost of ANY form of system interruption.

Lost Opportunities

What is the value of the time lost when the computer is down? Opportunity costs are the value of opportunities foregone. If you are unable to sell an item because the computer is down then this would be an opportunity foregone. Other examples might include loss of sales because customers lose confidence in your ability to forecast future market trends properly and introduce new products.

Real Costs

While the computer is down, you will be experiencing costs that are directly related to the resource (computer) or the event. This becomes painfully apparent when the computer is no longer avail- able, and you have a number of data entry personnel idle. You will have employees who are paid and bored, a double whammy! The computer itself also has a cost associated with it. Remember that you are still paying for hardware and software maintenance; in addition, the computer (if it still exists) is depreciating.

Consider the temporary loss of the accounts receivable and mailing lists. This could cause a severe cash flow problem in many companies. If your data is used for manufacturing planning, the temporary loss of data may result in production slides and material shortages. This translates to lost sales!

Finally, what is the value of your data? This should be the most obvious, explicit cost; however, it is often forgotten. A simple rule of thumb is that the value of the data is worth at least the cost of the machine. Often it is worth much more than that.

GENERAL DISASTER PROTECTION TECHNIQUES

Backup Procedure and Frequency

The best protection against disasters is consistent and intelligent system backup procedures. That is to say, you should have an established regular backup procedure, performed by operators who know what they are doing, with a media that is reliable. The frequency and method of your backups, whether partial or full, should be worked out with your HP Customer Engineer and is dependent on your system activity.

A good complete set of backup tapes is the ONLY way to guarantee that you may migrate to another site if necessary.

Backup Costs

With backups being such an important protection, why not fully backup the computer everyday? The answer is simple: backups are expensive. These costs can be broken down into several categories: labor, computer down time, and materials.

At our site, we estimate the time needed to backup data on magnetic tape (HP7970) at thirtyfive seconds per megabyte of files. This figure includes the time that our operators must spend changing tapes, cleaning the drive, and labeling the tapes. Therefore, if our system contains four hundred megabytes of data, the time needed for backup would be 228 minutes, or almost four hours.

When a full backup is in progress, it is best if the computer can be made as idle as possible so that as many files can be backed up as possible. This requires that either the computer be made idle during the work day or that you have operators running second-shift backups. At some HP3000 sites, the computer is in use nearly 24 hours a day. Backup time is essentially the same as down time.

Roughly 30 megabytes of data can be stored on one 2400 foot tape (HP7970). This means that our four hundred megabyte store would require about 14 tapes. The current price per tape is in the 20- to 25-dollar range. We would expect tapes for one full backup to cost us about $300.

Finally, how long do you hold your backup tapes? We rotate our backup tapes every two months. If we were to backup the system every night, then we would need storage for more than 500 tapes! The material investment would probably exceed $10,000. We have not included in our rough costs the amount spent on the paper backup listing, drive wear and tear, or the fact that after a few months you finally bite the bullet and order a 6250 bpi drive!

Obviously it is usually not feasible to perform a full backup every night. HP recognized this and gave us the partial backup ability. However, if your application uses a large (e.g., 300 megabyte) data base and virtually all data sets are modified every day, then a partial is not much faster than a full backup!

Okay, so what's the final tally? Well, the backup costs will be roughly:

```
Backup$ = Cost of down time + materials + cost of partials in between

where:
    cost of down time = labor for recovery + idle labor + lost
                        use of computer

    materials         = cost of tapes + cost of paper
                        + storage costs

    cost of partials  = cost of partial down time
                         + materials + labor
```

Transaction Logging

Since it is unreasonable and, in some cases impossible, to have a backup for the computer every few minutes another technique must be used. Such a technique is called transaction logging. The basic concept is to make a copy of changes to databases. This separate change file then can be used with a known good copy of a database to resynchronize the data in the event of a system interruption. Transaction logging allows, in effect, a continuous backup of the database to be made. Unfortunately, only transactions made on IMAGE databases are automatically logged by HP software. There is third party software available which will allow you to log transactions on KSAM and MPE files also.

There is no such thing as a free lunch; logging does have costs associated with it. The additional overhead for systems with sufficient memory is a 3% to 8% reduction in throughput. With logging you must have a place for the transaction copies to go. If this place is a tape drive, then you have the added cost of the drive dedicated to logging. If you are logging to disc, then you must allow sufficient disc space or periodically shut down the application and switch the log file.

Alternate Site (Mutual Backup) Agreements

Locate computer installations with similiar capabilities which could act as a backup site. There are several varieties of backup sites. Richard Gray, in an article in the March/April 1983 Interact, described three common forms. They are cold, warm, and hot backup sites.

Cold Sites

A cold backup site is generally a computer facility all ready to be used except it is missing the computer. This type of facility has a limited usefulness. Any company using a cold backup site should have an agreement with the hardware vendor so that the downed site has a high priority on new equipment shipment when needed. The cold site agreement is attractive because of the low capital cost; again, the disadvantage is that in the event of a total computer loss, the down time may be days, weeks or months.

In evaluating the cold site approach be sure to include the cost of the empty shell (room). Typically user occupancy (UO) costs run $5 to $15 per square foot per year. This means that a empty computer room that is 15 by 20 feet, and has a UO cost of $10 per square foot, could cost $3000 a year.

Warm Sites

A warm backup site is generally another computer installation which will make room for your application. A typical warm site is a computer service bureau. A warm site has a low capital cost and potentially shorter response time than cold sites; however, there are disadvantages. You may have compe- tition for the computer if the disaster is not confined to your business. In addition, the cost of the service can be quite high.

If your company has more than one system (at different locations) it may be possible to establish warm sites within your company by adding additional hardware and software, so that the systems appear to be mirror images.

Hot Sites

A hot backup site is generally a computer which is idle, just waiting for a disaster to happen. This type of backup site provides the fastest possible method to get up and going again. The obvious disadvantage is the high capital cost.

Service Agreements

HP provides a wide variety of service contracts. It is important that you match the service agreement with your needs, not your pocketbook. If during your risk analysis you have determined that your maximum allowable down time is four hours, then you should have, at the very least, a four-hour response contract with HP! Do not let the cost of service contracts be the driving force if you really need fast response.

Some things to consider when considering service agreements: guaranteed up-time, hardware reliability, guaranteed response time, and, lastly, costs.

Site Protection

There is a lot of truth to the saying that "an ounce of protection is worth a pound of cure". Be sure to build fire protection into any computer facilities. If you are storing your backup media at the same site, look into the purchase of an UL-approved fireproof data safe. Remember also that in many cases the damage is not done by the fire itself, but by smoke or water. Fire extinguishers should be the HALON type only. In addition, keep a supply of fire-retardant plastic available. In a recent fire at the Department of Health and Social Services in the state of Washington, an alert disaster team was able to save the disc drives and databases from serious water damage by covering the computer system with sheets of fire-retardant plastic.

If your business resides in a flood plain, then you should place the computer on as high a floor as possible. Backup tapes should be kept off site, in a MUCH higher place. Finally, during the rainy season, it may be worth altering your backup cycle to cover the increased risk.

Is your computer room earthquake-proof? Although most disc drives would survive a moderate earthquake, we often add extra hazards by placing tall bookcases in the same room. Then, when an earthquake strikes, the case falls and destroys the drive. If you must have the book case in the same room, anchor it!

Recovery Costs

What does it cost to recover after a disaster? This simple question is not easily answered, since the recovery cost is a function of a number of variables. For instance, how bad was the damage? How long will it take you to assess the damage to the data base and associated files? When will the computer be available?

You will never be able to predict exactly the time and cost to recover, but you can learn to arrive at good "ball park" figures. This can only be accomplished after you have gained enough experience. The needed experience is obtained during the day-to-day operations and by running practice drills.

Getting It Right

All of us have experienced a condition known as system failure. The steps needed to recover after a failure and the time spent recovering are perhaps the most valuable piece of information when designing your disaster plan.

Disaster drills help fill the gap in the practical experience you have gained in your day to day computer operations. Remember, one of the simple corollaries to Murphy's law is: "If anything can go wrong, it will do so at the worst possible time." No disaster plan can anticipate all possible problems, but drills can help pin-point the obvious mistakes and omissions.

When you hold disaster drills, suprise as many people on the recovery team as possible. Try to make the drill as realistic as possible. Now that does not mean you have to have smoke coming out of the windows and the sprinklers going, but it does mean that if the drill is for a fire which has completely destroyed the facility, no one should go to his/her desk for notes, etc. Assign one person on the team to take notes on the events during the drill and what material was needed or missing. After the drill be sure to review the notes with all members of the team. Finally, if you found any mistakes in the plan BE SURE to update it!

Computer Insurance

Most insurance companies can provide insurance against loss of facilities or loss of data. Consider some form of insurance as a supplement to your disaster plan. But before signing on the dotted line, carefully check any insurance agreement for cost, the deductibles, insurance coverage, and, most importantly, exclusions!

The Balancing Act

By now you have determined the cost for each level of a disaster to your system. The trick is to devise protection schemes which will provide sufficient protection at less cost than the actual disaster.

THE PLAN

Remember, when a disaster does occur, your best system analyst may be on vacation in some inaccessible portion of the world. Therefore do not count on his knowledge! Instead, make sure that your disaster plan addresses what and which records are vital for your company, what steps may be taken to recover the data, salvage operations (if necessary), alternate equipment, and who will be doing the work! Here is a simple outline for a recovery plan for your site.

A) INTRODUCTION

B) DEFINITIONS

C) SCOPE OF THE PLAN

D) STATEMENT OF LOSS

1) OTHER SYSTEMS DEPENDENT
2) LEVELS OF DISASTER

E) ALTERNATE SITE REQUIREMENTS

1) GENERAL SITE INFO
2) CURRENT SITE LOCATION
3) RECOVERY TEAM
4) HAZARD PROCTECTION
5) SECURITY PROCTECTION
6) BACKUP PROCEDURES
7) CRITICAL DATA AND PROGRAMS
8) SPECIAL FORMS
9) HOURS NEEDED
10) DATA COMMUNICATION
11) MINIMUM HARDWARE
12) SPECIAL SOFTWARE
13) DOCUMENTION
14) TRAINING

F) RECOVERY TEAMS

1) LEVEL 1 RECOVERY CREW
2) LEVEL 2 RECOVERY CREW
3) LEVEL 3 RECOVERY CREW

H) RECOVERY PROCEDURE

1) RECOVERY TEAM MEETING LOCATION
2) LEVEL 1 PROCEDURE
3) LEVEL 2 PROCEDURE
4) LEVEL 3 PROCEDURE

I) POST-RECOVERY PROCEDURE

J) ALTERNATE SITE AGREEMENT

K) PLAN FOR UPDATING THE DISASTER PLAN

TOOLS MAKE IT EASIER

The following is a list of tools which are useful in planning and managing a disaster/recovery plan.

Backup & Sysdump Programs

Hewlett-Packard provides a backup procedure called SYSDUMP. Using the SYSDUMP facility you can perform either full backups or partial backups depending on how you answer the questions. If you wish to automate this backup process, then look into BACKUP and SYSDUMP from the contributed library.

BACKUP - Switches the system log file, then builds a stream job for the backup, and finally streams it.

SYSDUMP -Maintains the date of the last full backup and is useful in determining the frequency of partial versus full backups.

Account Structure Programs

An absolute necessity for any type of disaster plan is a current Accounting Structure list. Use this list to recreate the accounting structure after a NULL reload or to migrate your application to another computer. Several very intelligent users of the HP3000 quickly discovered that maintaining the Account Structure list manually was not only tiresome, often it was never done. Why not let the computer do the work? Well, here are several programs which do just that.

ACCSTRUC - This program creates a series of stream files which can be used to re-create the accounting structure from scratch.

ACCTGEN - This program creates a job file which can be used to restore all users, groups, and accounts to a null system.

BACCT - This program creates a job file which can be used to restore all users, groups, and accounts to a null system. In addition, it provides general reports on file access and user capabilities.

BULDACCT - This program creates a copy of the accounting structure.

DIRK - This is a general purpose directory-query program. DIRK extracts information about files, users, groups, and accounts by wild cards, and such items as file code, last access date, size, etc. One of the handy features of DIRK is its ablitiy to generate a stream job for a user account or the whole system.

UDC Utility Programs

Almost every HP3000 installation has customized the MPE commands with user definable commands (UDCs). These custom commands are often interspersed in critical stream jobs. For this reason it is important to keep a list of the UDC commands and files with any disaster plan. Here are some programs from the contributed library which will help out:

LISTUDC - This program will list the UDCs in effect for the system. The report generated is a handy reference to have in the event of a disaster recovery.

UDCLIST - This program generates a list which shows which UDCs are in effect.

CMDANAL - This program generates a list of UDCs which are in effect for the system.

Stream Job Utility Programs

It is entirely possible that the sole cause of a disaster on your system, is an unwanted visit by a computer HACKER. Currently the stream facility that HP has provided requires that passwords be inserted in the stream file. This poses a very severe security risk. If you must move to an alternate site, you do not want to be bothered with changing passwords on all of your files in addition to trying to recover lost data! Again the user community has come through and developed a number of very useful programs. These include:

DOJOB - DOJOB lets you build a job card template, then when a job is streamed, DOJOB looks up the necessary passwords and inserts them.

JES - JES allows you to maintain a schedule of when jobs are to run. JES supports job card templates and allows parameters to be inserted.

JOBQUE - JOBQUE lets you set up stream files to be submitted at any hour of any day. Other features include user notification if the job was not launched when requested (system was down, etc). In addition, JOBQUE supports a template which allows the users to build the stream jobs without including the passwords in their files.

SLEEPER - streams jobs, executes MPE commands, and runs programs at desired intervals.

SLS - An friendly interactive facility which streams jobs immediately, at desired times, and has job templates.

STREAMER - streams a job that passes system and site security and provides the passwords for the job cards.

MPE Data Recovery Utilities

The tools listed in this section should only be used by very experienced personnel.

SADUTIL - Stand alone diagnostic that allows file recovery. This utility is HP supported.

RECOVER2 - Restores files from the tape created by SADUTIL.

STAN - Reads the directory from the backup tapes and display the file creator information.

DISKED2 - Disk Edit utility program supported by HP.

Data Base Recovery Tools

HP provides four useful tools to protect your IMAGE data bases. These are DBSTORE, DBUNLOAD, DBLOAD, and DBRECOV. In addition, there are a number of tools available from third-party software firms, such as ADAGER from Adager, S.A., RECOVERY/3000 from Abacus, and SUPRTOOL from Robelle, to name a few.

STREAM JOBS MAKE EASY WORK

Simplify your disaster recovery procedures by establishing a couple of simple stream jobs. These jobs then can be used to recreate and restore the important files from your system on an alternate system, when necessary.

Stream Jobs For Accounting Structure

The following stream job uses the contributed library program 'DIRK' to create a file which can be used to recreate the complete accounting structure for your system.

```
!JOB ACCOUNT,<<USER.ACCT>>;OUTCLASS=,1
!COMMENT  ************************************************
!COMMENT  *  This job creates a copy of the WIDGETS accounting*
!COMMENT  *  structure and places the output in a file called *
!COMMENT  *  WIDGETS.DATA   The file is used to recreate the  *
!COMMENT  *  the WIDGETS application at another site if the   *
!COMMENT  *  of a DISASTOR RECOVERY PLAN is invoked.          *
!COMMENT  *                                                  *
!COMMENT  *  This job is stream every night by JOBQUE.  The  *
!COMMENT  *  logon user is MANAGER.SYS.  We have a special    *
!COMMENT  *  file group call ACCOUNT.  When                  *
!COMMENT  *  we migrate to another site, we only need to add  *
!COMMENT  *  the ACCOUNT group to the host computer, and      *
!COMMENT  *  restore WIDGET1  into the group.  Then we stream *
!COMMENT  *  stream WIDGET1.ACCOUNT.SYS.  This will build our *
!COMMENT  *  needed accounting structure.                     *
!COMMENT  ************************************************
!RUN DIRK.PUB.TECH
KILL WIDGET1.ACCOUNT
BUILD WIDGET1.ACCOUNT;REC=-72,,F,ASCII;DISC=3000,32,8
OUT WIDGET1.ACCOUNT
NEWA @;OKPASS
EXIT
```

```
!COMMENT   We are not worried about the file security for WIDGET1
!COMMENT    because we created the file group with CREATOR only
!COMMENT     file security rules!
!EOJ
```

We use the contributed library program JOBQUE to submit this job every night. This way a current copy of the accounting structure is always included with our partial and full backup tapes.

Stream Jobs for Reloading

The following job is an example of the accounting structure file which was created by the previous job.

```
!JOB NEWACCT,MANAGER.SYS;OUTCLASS=,1
!CONTINUE
!NEWACCT WIDGETS,MANAGER
!ALTACCT WIDGETS&
!STREAM ,#
#JOB NEW,MANAGER.WIDGETS,PUB;OUTCLASS=,1
#CONTINUE
#NEWGROUP PUB
#ALTGROUP PUB&
#;ACCESS=(R,X:ANY;A,W,L,S:AL;A,W,L,S:GU)
#
#CONTINUE
#NEWUSER MANAGER
#ALTUSER MANAGER;HOME=PUB&
#;CAP=AM,AL,GL,ND,SF,BA,IA,LG
#
#CONTINUE
#NEWUSER WIDGETS
#ALTUSER WIDGETS&
#;CAP=ND,SF,BA,IA,LG
#
#EOJ
!EOJ
```

To rebuild the accounting structure on a new host machine after a disaster, we simply restore the WIDGET1 file, and stream it. The passwords will be exactly the same as before the disaster. The next step is to restore the files from the WIDGET account, and begin our database recovery procedure.

Recovery

The exact recovery steps that you must follow depend on the type of disaster you experienced. In the case of the common system failure you can simply follow the standard procedures setup by HP. In the more complicated failure resulting in a disc crash, you may need to use some of the data recovery techniques discussed by Goertz and Beasley [1],[3].

Be sure to check your database for structural damage after any major disaster. It is also a wise policy to have a specialized program which checks the database for semantic errors.

After the Crisis

After the disaster is over, and you have successfully revived your application, it is time to review the steps taken during the crisis and incorporate any important new steps into your disaster plan.

Remember that you may have minor mistakes in your database for a number of weeks following any disaster. Make sure that your employees are aware of this fact, and that they report any glitches to the data processing department for investigation.

SUMMARY

In this paper we have discussed ways to place a value on the loss of your computing facility, the time to recover, and finally some of the tools that are available in the contributed library. As can be seen the developing a disaster and recovery plan is not a simple task. It requires careful thought and dedication on the part of your data processing department.

Appendix A is a check list of questions that can be used to determine if you have an adequate disaster plan. Appendix B is an example of a disaster plan for the ACME WIDGET company. Appendix C contains excerpts of the recovery plan for ACME.

Appendix A. Disaster and Recovery Check List

A) GENERAL PLANS

Do you have a disaster plan? Is a copy of the disaster recovery plan stored off site? Do you have a plan to update the disaster plan? Have you identified the disaster recovery team personnel? Have you identified an alternate meeting location? Is someone designated to keep track of the events during the recovery?

B) BACKUPS

Is the system backed up? What is the backup media? Do you periodically check your backup media? Do you maintain a list of files on the backups? Where is your backup stored? How long is your backup copy archived? What insures that a backup is done?

C) FACILITIES

How fire-sensitive and combustible is your computer room? Do you store unrelated materials in the computer room? Where is your excess paper stored? What type of smoke and fire detectors are installed in your computer room? Is there an emergency shutdown for the computer? Do you have a fire extinguisher and are your personnel trained to use it? Do you have fire retardant plastic which can be used to cover the computer and reduce water damage? Is the area under your false floor kept clear? What steps have been taken to protect the computer from brownouts and power surges? Is the building structurally sound? Can it withstand an earthquake? Are bookcases isolated so they will not damage the computer in the event of earthquakes? Are electrical junctions boxes under the false floor protected from water damage? What type of losses are covered by your insurance?

D) SECURITY

Is access to the computer room controlled? Is the computer room locked? How often is the lock changed? How many levels of LOGON passwords are required? Have you changed the passwords for the vendor-installed accounts? (Adager, Robelle, Quasar, etc.) How frequently do you change passwords? Do you monitor invalid logon attempts? How is access to your databases controlled? How do you handle security violations?

E) ALTERNATE SITE

Have you identified a possible alternate site? (cold, warm or hot?) Is the hardware and software compatible? Do you have a written agreement? Have you tested your recovery procedure on the alternate machine?

Appendix B. Sample Disaster Recovery Plan

1.0 INTRODUCTION

The WIDGETS application provides inventory visibility, on-line order processing, quality assurance (QA), materials requirement planning (MRP), marketing and finance information for the ACME WIDGET Company.

The WIDGETS application is critical to the conduct of ACME business. These procedures insure continued critical system operations and planned system recovery in the event of a local disaster.

2.0 DEFINITIONS

Disaster - a system interruption, local to the WIDGETS central processing system which does one or more of the following:

a. Disables the WIDGET system for more than 8 hours.

b. Permanently disables all computing hardware at the computer site.

c. Destroys critical data stored on tape or disk media at the computer site.

d. Disables system manager personnel at the computer site.

3.0 SCOPE OF DISASTER PLAN

This plan covers all applications and data which are used on the WIDGET1 computer. Excluded from the plan are the programming and new software projects which are being developed on the WIDGET2 computer. Peripheral support equipment (terminals) are excluded because we can satisfactorily meet our users' needs through priority reallocation of hardware and personnel.

4.0 WIDGETS IMPACT STATEMENT FOR LOSS OF SYSTEM

4.1 Other System Dependencies

The WIDGET1 computer does not pass any data to any other systems.

4.2 Stage 1

The WIDGET1 computer can be down for up to 24 hours with little or no impact on any external application. Down-time of up to 24 hours will require the system manager, ACME site manager, and a limited number of other personnel to be on hand to recover the programs and data base.

4.3 Stage 2

For down times of 24 hours to approximately 72 hours the WIDGETS data and programs can be recovered by providing additional manpower or overtime. We would expect to lose sales and a one week delay in shipment of our best selling WIDGET. Based on previous experiences, we can expect some minor complaints from customers, but no long term adverse effects on the company.

4.4 Stage 3

If the WIDGETS programs and data base are not available after 72 hours (3 days), the WIDGET recovery crew can no longer absorb the extra workload. At this point, there is a significant risk of loss of capital equipment, financial data, and MRP data. In addition, after 3 days it is probable that a number of our best customers would switch to the AAA Gadget Company. This loss of sales could have a significant impact on the future sales of spare parts, service, and maintenance.

5.0 WIDGETS BACK-UP REQUIREMENTS DATA

5.1 General Site Information

The WIDGETS system is comprised of an HP 3000 Series 30 computer facility and eleven remote peripheral support areas located throughout the greater Two Dot area.

5.1.1 Site Location

The WIDGETS site which is the subject of this disaster plan is located within ACME, Building 1, Room 2.

5.1.2 Widget System Contacts

ACME President - Honest Abe,
Work Phone 555-1234, Home Phone 800-1600

WIDGETS Operating Staff - Operations Manager, Jim Snodgrass
Work Phone 555-5693, Home Phone 556-8716

- Alternate, Paul S. Simon
Work Phone 555-3747, Home Phone 800-8263

- Manufacturing Manager, Issac S. Best
Work Phone 555-8732, Home Phone 556-4239

- Quality Assurance, Verl Y. Accurix
Work Phone 555-7777, Home Phone 556-9673

- Finance, Buck M. Green
Work Phone 555-8723, Home Phone 800-9623

- Clerical, Meg Good
Work Phone 555-9123, Home Phone 556-9673

5.1.3 Hazard Protection

Fire Protection - Conventional Overhead water Sprinkler System

Environment - Building Air Conditioning with Emergency Shut Down Switch

Electrical - Conventional Commercial Power with Emergency Shut Down Switch

Communications - Conventional voice grade telephone lines

5.1.4 Security System

Limited Access to Area - Simplex Lock

Logon Password - WIDGET System Access

5.1.5 Backup Frequency and Log Cycles

The WIDGETS application is fully backed up on Wednesday evenings. The backup tapes are maintained offsite (Building 2). In addition to weekly backups the WIDGET data base is stored, and a transaction log file is maintained. The log tapes are stored in the same room as the computer.

5.2 Critical Requirements, Applications, and Data

5.2.1 Application

The WIDGETS provides the following critical applications:

- Visibility on the location and use of a dynamic inventory of Widgets (4000 new widgets produced weekly).
- Random sample system for quality assurance.
- Market forecasting for sales
- Finance invoicing and payroll.
- Materials requirement planning.
- Production scheduling.

5.2.2 Interface

Critical interface with other systems include the following:

5.2.3 Programs, Data Bases, and Files

5.2.3.1 Critical Programs

All programs in the PROG group of the WIDGETS account are required for the application to run.

5.2.3.2 Critical MPE Files

All MPE files in the DATA, JOBS, REPORTS, and QUERY groups of the WIDGETS account are required for the applications to run.

5.2.3.3 Critical Vendor Software

The application programs require HP's MPEIV operating system with a version of Q-MIT or later. No third party vendor software is required.

5.3 Specific Forms

The type and quantity of paper required for proper WIDGETS operation are documented in the WIDGETS OPERATOR GUIDE.

5.4 System Usage and Hours of Operation

5.4.1 Normal Hours

The WIDGETS computer is used 24 hours a day by either on-line programs or background batch jobs.

5.4.2 Critical Hours

The WIDGETS computer must be available from 6:30 AM to 4:30 PM Monday through Friday.

If transaction logging is used, then the application and data bases need only be backed up on a weekly basis. Based on previous history the time needed to perform these backups is 4 hours once a week. If transaction logging is not used then this requirement is extended to 2 hours 5 days a week after 4:30 PM.

The WIDGETS application must have the computer exclusively the third weekend of every month for 20 hours. This time is used for billing and payroll.

5.5 Number of Ports Required

To continue operating, the WIDGETS programs require at least seven phone lines into the computer. There will be one phone line assigned to each of the following groups:

Manufacturing Quality Assurance Finance Order Processing WIDGETS data processing staff ACME WIDGETS president

The remaining phone line will be a floater; that is, anybody can logon to this port and use the computer for up to 15 minutes at a time.

5.6 Minimum Hardware Required for the WIDGET System is:

HP 3000 Model 30 or newer 300,000 sectors free disc space (75 megabytes) 7 1200-baud Bell 212A compatible phone lines 1 lineprinter 400 LPM minimum 1 tape drive for backups and logging. If the drive is not available for logging, another 50,000 sectors of disc space is required.

5.7 Documentation

See the WIDGETS operator and user manuals. After the WIDGETS system has been moved to the alternate site, extra copies of the user and operator manuals may be obtained by streaming USERGUID.DOCUMN and OPGUIDE.DOCUMNT.

5.8 Training

At least three of the recovery team members should have taken the system supervisor class from HP. In addition these members should also review the system supervisor material in the HP manuals at least once a month.

6.0 RECOVERY TEAMS

The recovery personnel fall into three categories: management, operating, and supportive. The management team will consist of the program manager, the ACME functional custodian, and the WIDGETS operation manager. The purpose of the management team is to assure that no unnecessary road blocks develop during the disaster recovery. The operating team will consist of the WIDGETS operation manager, the system manager, a representative from HP, and the WIDGETS analyst. The function of this team is to implement the recovery procedure as fast as possible. The final team is support personnel. The support team will consist of the remote site dependent staff and office personnel from the WIDGETS site.

7.0 RECOVERY PROCEDURE

The recovery procedure that will be followed is documented in the WIDGETS operators guide section 5.0 (Appendix C of this paper).

8.0 AFTER RECOVERY FOLLOW UP

After the recovery has been completed, stream the job called ERRCHECK.JOB to check the data base for structual damage and semantic correctness. The error checking job needs about ten hours to run to completion based on previous experience for the WIDGET system.

The section managers should also remind their employees that they can expect to find minor mistakes in the data. These errors should be reported to the WIDGET data processing staff on the appropiate forms so that they can be investigated and corrected.

If the disaster was a level 3, then all section managers create special teams to "inventory" their areas of responsibility. The WIDGETS data processing staff will assist in generating the needed audit reports.

Finally there will be a weekly meeting of the recovery team to discuss the status of the after-recovery inventories. The team will continue to meet and address any problems that arise until satisfied that the crisis is over.

After the crisis is over the WIDGETS operation manager will review the notes taken during the recovery procedure and make any necessary corrections in the disaster plan.

Appendix C. Excerpts from WIDGETS Disaster and Recovery Procedures

1.6 Emergency Shut Down

This section will cover two basic types of emergencies, a fire in the Computer Room and a disaster such

as Ash Fallout or similar incident that is predicted ahead of time.

1.6.1 Fire

In the event of a fire in the computer room there is a RED Emergency Disconnect Switch which is located just inside of the entry door on your LEFT. (It is just above the Line Printer). PULL the Center Section DOWN as the instructions on the switch direct. Then shut the door (make sure that it is UNLOCKED) and evacuate the premises. PLEASE BE SURE THAT YOU REALLY WANT TO USE THIS SWITCH. The only way that power can be restored to the room is by Facilities replacing the unit. IT IS NOT AN "OFF/ON" TYPE SWITCH.

1.6.2 Disaster

In the case of a disaster, such as an Ash Fallout, the following procedures should be used:

A - Get everyone off the system.

B - BACKUP the system. (See Chapters on System Backup, Data Base Store, and Weekly Backup Procedure.)

C - When the backup is done, shut the system down in accord- ance with the procedures in Paragraph 1.5.1 of this document.

D - Cover all equipment with the Fire Retardent Visquine which is located behind the door of the room. Please tape the Visquine to the bottom of all equipment cabinets.

E - Place all the tapes in the Tape Cabinet in plastic bags and replace in the cabinet. Be sure to shut all of the cabinet doors.

F - Secure the entry door to the Computer Room.

5.0 SYSTEM CRASH RECOVERY PROCEDURE

There are several different types of errors and levels of severity when CRASHES of the system and the data base are involved. If you are 100% sure that there were no data bases open*, then the data base recovery procedures listed below can be skipped. It would only be necessary to start a new log tape; however, if the FAILURE occurs when the system is active (users are on the system) it is a diff- erent story. *NOTE - When in doubt go through the full recovery procedure!

IN ANY CASE BE SURE THAT THE TAPE DRIVE IS FUNCTIONAL AND THAT THE TRANSACTION TAPES HAVE NOT BEEN CORRUPTED BY THE FAILURE.

The following section covers who to notify and how to determine the type of crash.

5.1. WHO TO CALL

Contact all users. Ask them to save all information on their last several transactions. Also get ANY INFORMATION displayed on their terminals (have them copy it EXACTLY as printed on the terminal). Tell them to shut down their terminals (logging off will not be necessary as the system has stopped). Inform the users that "WE" will notify them when they can get back on the system.

5.1.1 System Crashes Versus Transaction Logging Problems

5.1.1.1 Tape or Logging Failures

If the system has displayed a message which says that it is unable to write or use the transaction log, then the failure involves transaction logging only. The message will be similiar to the following:

ERROR WHILE WRITING TO USER LOGGING FILE LOGTAP (ULOGERR 7)

If such an error occurs then SKIP SECTIONS 5.2 THRU 5.6 AND

*** GO TO PARAGRAPH 5.6.5 ***

5.1.1.2 System Failures

In the event of a system crash that brings a halt to the HP 3000 (BE SURE THAT IT IS A HALT DUE TO A SYSTEM FAILURE), proceed with the following sequence of events.

Log the crash in the "GOLD" System Book and indicate any system failure messages that appeared on the console.

5.2 Memory Dump and Log Recovery

The MEMORY DUMP stores everything that was in the main memory of the computer to a designated device (Mag Tape or Floppy Disc). This data is later printed out on the line printer and used for analysis of the failure. Perform the dump in the following manner:

A. Rewind and remove the Log Tape from the Tape Drive, then mount a fresh magnetic tape and place the drive ON-LINE.

B. Press the "MEM DUMP" on the Front Panel, when the MEMORY DUMP aborts enter the following commands in from the console. You should have a ">" prompt.

>DUMPDEV 41,0 >DUMP

At this point the Software Dump Facility (SDF) will begin a serial execution of the file SDFCOM which is located on the system disc. Then the following message will appear on the console:

SOFTWARE DUMP FACILITY
(VER XX.XX/XX)

When the HALT light comes on check and make sure that the tape drive is ON-LINE.

Press the RUN button on the front panel. The system will then dump everything that is in the memory to the tape drive and display:

DUMP COMMENT. IF YOU WISH TO CONTINUE WITH A WARMSTART COMMENT. THEN PRESS RUN. HALT

C. Press the RUN button on the front panel again. This will let the system begin the WARMSTART procedure. When the "RUN/HALT" field displays "RUN" hit the Return Key.

D. Reply to any questions that appear on the console (i.e., day/ month/year and time of day).

E. The system will issue a "Welcome" message and automatically logs on "OPERATOR.SYS;HIPRI". It will then print a series of lines of data depending upon the system configuration.

F. At this point IDLEJOB will ask if you want to proceed with the startup procedure. PLEASE ANSWER NO.

G. Remount the Transaction Log Tape and make any necessary replies to the console. Do a "SHOWLOGSTATUS". This should tell you that transaction logging is recovering. When the system indicates that logging has been recovered (or failed to recover), you may proceed.

H. If the system has indicated that Transaction Loging has recovered, use the console and do a "LOG WIDGTLOG,STOP".

FOR ADDITIONAL INFORMATION ON "MEMORY DUMPS" REFER TO THE SECTION ENTITLED "SOFTWARE DUMP FACILITY" IN THE HP 3000/33 OPERATORS MANUAL.

5.4 Saving the Transaction Log

In order to preserve the Log Tape for future analysis to determine if any data was lost, we want to put a copy of the Log Tape on the disc and get a listing from the line printer. This is done in the following manner:

A. From an Office terminal (if none were on when the system crashed you can logon by using ":HELLO ACME.WIDGETS;HIPRI") do a ":STREAM LOG.REPORT". This stream job transfers the log information to a disc and also lists limited information on the line printer.

   BE SURE TO EXAMINE THE LISTING! ESPECIALLY LOOK FOR THE FOLLOWING MESSAGE:

   BAD LOGID# nnnn TRANSACTION LOG IS SUSPECT!

   Then ..... please ..... DO NOT USE THIS LOG FOR A DATA BASE RECOVERY!

B. When the listing comes off of the line printer remove the Transaction Log Tape from the tape drive and proceed with the next step.

5.5 Rebuilding the Database

When there are several different databases in use at the time of a system crash, each database must be rebuilt independently. As we are currently using only the WIDGET database it is the example that we will use. The following procedures will bring your data- base up to date with the latest Data Base Store in the Tape Library.

A. From an office terminal enter ":DBUTIL" (or :RUN DBUTIL.PUB.SYS), the terminal will display the prompt: >>enter PURGE WIDGET When the system has finished your office terminal will display the following: Data Base WIDGET has been Purged >>enter "EXIT"

B. Mount the latest DBSTORE Tape on the tape drive. Place it on-line (NO WRITE RING PLEASE) and proceed.

C. From the office Terminal

   :RUN DBRESTOR.PUB.SYS

   The terminal will respond with WHICH DATA BASE? WIDGET

   Go back to the console and make your reply (REPLY (PIN#), 7)

   When the database has been restored the terminal will display "DATA BASE RESTORED" (this will take approximately one minute for each megabyte of database or about 40 minutes for WIDGET), and then display: DATA BASE RESTORED

D. When the restore is done remove that tape (hang it up with the other tapes). Then remount the log tape that was running when the crash occurred (NO WRITE RING PLEASE) and put the tape drive on line.

5.6 Recovering the Database

When the restore is completed you must RECOVER the database by using the Transaction Log Tape to the point it was as of the system failure using the following procedures.

```
A.  From your Office terminal enter ":DBUTIL" (or :RUN
    DBUTIL.PUB.SYS), the terminal will display the prompt ">>".
    >> enter ENABLE WIDGET FOR RECOVERY
         RECOVERY IS ENABLED.
    >>DISABLE WIDGET FOR ACCESS
         ACCESS IS DISABLED.
    >>ENABLE WIDGET FOR LOGGING
         LOGGING IS ENABLED. (it may show "Data base already
         ENABLED for LOGGING")
    >>SET WIDGET LOGID=WIDGTLOG
         PASSWORD:  (USE THE RETURN KEY)
         LOGID:  WIDGTLOG IS VALID
                 PASSWORD IS CORRECT
    >>EXIT
```

B. From your Office terminal enter ":RUN DBRECOV.PUB.SYS"
the terminal will display the prompt ">".
>enter RECOVER WIDGET
DATABASE WIDGET LAST DBSTORED XXXXXXX (The system will give the day, date and time - it is called a "time stamp"). If this is not the same as the date of the DBSTORE tape - ABORT the job.
>enter RUN

5.7 Storing the New Data Base

Now that the data base has been brought up to the same config- uration that it was when the system failure occured, we should make a new DBSTORE tape for protection. Mount a scratch tape on the tape drive and put it "On-Line"

A. From your office terminal do the following: :RUN DBSTORE.PUB.SYS (You may use DBSTORE alone) WHICH DATA BASE? enter WIDGET Then go back to the console and make the appropriate reply. When the DBSTORE is completed (approximately 20 minutes) your terminal will inform you that the job is complete. Remove the tape from the tape drive, take the write ring out, and hang the tape up in the tape cabinet.

B. After the DBSTORE you should shut the system down, then bring it back up with a "COOLSTART". This time when the system console asks if you want to proceed with "Start Up" procedures answer "YES". This is the only way to get IDLEUTIL running again. (Remember, when we started it up before we didn't start IDLEJOB.)

5.8 Start Transaction Logging

Mount a new tape (small reel please) on the tape drive and put it on line. From the Console do the following:

:LOG WIDGTLOG,START
/VOLUME ID FOR TAPE (MAX CHARS = 6)?
REPLY (PIN#),LOGTAP
/MOUNT TAPE VOLUME LOGTAP (MAX CHARS = 2)?
REPLY (PIN#), 7

The system will respond by saying that transaction logging is running.

A. Before proceeding any further you should allocate the programs as outlined in Paragraph 5.9.

B. Go back to the Console and do an "ALLOWLOGON".

This will set all the fences at the proper position; however, do ":SHOWOUT". If the out-fence is above 5 do -

:OUTFENCE 1

5.9 SYSTEM IS UP

Call the users and tell them they can log on to the system. Get a cup of coffee and relax - the panic is over, and the job is done. Please note, I may have left some of the responses from the system out of this writeup; however, all of your inputs and required replies are shown. Any system responses not shown are basically insignificant unless you get an error message, THEN CALL FOR HELP.

Bibliography

[1] Beasley, David, "HP3000 Data Recovery", Proceedings 1983 International Meeting HP3000 IUG, Montreal, Quebec, Canada April 24-29

[2] Gaade, R.P.R., "Picking Up the Pieces", Datamation, January 1980

[3] Goertz, Jason, "System Disaster Recovery: Tips and Techniques", Proceedings 1982 International Meeting HP3000 IUG, San Antonio, Texas, February 28 - March 5

[4] Gray, Richard, "Disaster-Recovery Strategies: A Back Door", Interact, March/April 1983

[5] Green, Robert M, and Heidner, Dennis, "Transaction Logging Tips", Proceedings 1983 International Meeting HP3000 IUG, Montreal, Quebec, Canada, April 24 - 29

[6] Heidner, Dennis. "Transaction Logging and its Uses", Proceedings 1982 International Meeting HP3000 IUG, San Antonio, Texas, February 28 - March 5

[7] Lazar, C.W., "HP 3000 Security/Risk Management", Proceedings 1981 International Meeting HP3000 IUG, Berlin, Germany, October 5-9

[8] Lloyd, Ben, "At the Movies", Interact, October 1983

[9] Lloyd, Ben, "Sane System Maintenance", Interact, September 1983

[10] Lohman, Guy and Muckstadt, John, "Optimal Policy for Batch Operations: Backup, Checkpointing, Reorganization and Updating", ACM Transactions on DataBase Systems, Vol. 2 No. 3, September 1977, pages 209-222

[11] Sardinas, Joseph, Burch, John G., and Asebrook, Richard, "EDP AUDITING: A PRIMER", Copyright 1981, John Wiley & Sons

[12] Sayani, Hasan, "Restart and Recovery in a Transaction-Oriented Information Processing System", ACM Workshop on Data Description Access and Control, May 1974

[13] Wise, Gerald, "Utopia", Hewlett-Packard Bonneville Regional Users Group Newsletter, February/April 1982

Dennis L. Heidner received the BSEE degree from Montana State University, Bozeman, Montana. He joined the Boeing Aerospace Company (BAC) in 1978 on a special project to review current techniques for management of general-purpose electronic test equipment. Based in part on this review of management methods used throughout industry, the BAC Test Equipment Management group received approval to buy a Hewlett-Packard HP3000 computer. Mr. Heidner was responsible for the system requirements planning, design and program implementation. In May 1980, the Test Equipment Inventory Management System became functional. Mr. Heidner has written "Transaction Logging and Its Uses", presented at the 1982 HP IUG in San Antonio, Texas. He was the co-author of two papers, "Transaction Logging Tips" and "IMAGE/3000 Performance Planning and Testing", which were presented at the 1983 HP IUG meeting in Montreal, Quebec, Canada. Mr. Heidner is a member of the IEEE Computer Society and the Association for Computing Machinery.

Optimizing the Operations Environment through the use of Manufacturing Techniques

by Victor Rozek
and
Sheila Dummer;
Hewlett-Packard

Two percent of sales, a loose standard for estimating data processing budgets, can easily run into the millions for a mid sized company. The business end of the budget is generally reserved for massive hardware purchases. Applications software packages and the escalating salaries of the programming staff. Almost as an afterthought, what can not be spent elsewhere is allocated to the support of computer operations.

Computer operations is the Rodney Dangerfield of data processing. Operators are generally entry level personnel, marginally trained and poorly compensated. Often working alone during off-shift hours. They are denied access to the high priced expertise of those who cause most of their problems. It is fashionable to blame them for everything from sloppy programming to hardware failures.

The theory is that anyone can run a program. Quite so, but even the most demanding gentle reader would have difficulty running 50 to 200 programs while keeping track of schedule requirements, sequence, job dependency, prompts, number of copies, special forms, restart and recovery procedures and media requirements. Yet that is precisely what many operators are asked to do with little more than a 3X5 card with scribbled instructions to guide them.

Yet so much of the success of a data processing department depends on the efficiency of its operations staff. The finest programming efforts are rendered meaningless if the G/L is closed before the receivables are posted; if people can't get their work done because there aren't enough copies of a critical report, if manufacturing must quesstimate requirements because an 18 hour MRP blew up and there were insufficient restart instructions.

This paper is submitted in the hope of providing help for the beleaguered operations staff. Areas of concern and neglect will be defined and solutions presented based on a model of manufacturing techniques. We will attempt to identify methods for optimizing throughput in a batch environnment, and providing timely and accurate output while generating a minimum of friction between the applications, operations and user communities.

Several assumptions are made: notably that we are all pregnant with Packard and therefore one or more HP3000 systems are being used in a busjness environment. Batch processing is defined as a combination of updates and reports from multiple data bases, typically run during evenings and weekends. During batch processing, time share users are at a minimum. Further, we strongly suspect that the HP3000 was designed to be a multiprocessing system, optimized for the performance of multiple simultaneous tasks - in spite of your experience to the contrary.

The first step in developing a manufacturing to data processing analogy is to examine briefly the functional organization of each group. Figure 1 illustrates a typical manufacturing organization and shows a comparable functional organization for a data processing department. Note that these are functional organizations; in small data processing groups several functions may be performed by the same individual. Overlooking any function is, however, a common cause of ineffective operations.

Using the data processing to manufacturing analogy, the first concern is capacity planning. Generally a sales force forecasts sales and on

the basis of these projections, production planning estimates manpower, equipment and material requirements. In our model, business systems analysis, with input from the user community, forecast hardware requirements including disc space, CPU processing time and printed output, based on projected growth in existing application systems and new applications under development. The operations and/or technical support group plan new hardware purchases and schedule installation and training.

Formalizing this process and matching it to the company's budget insures that aquisition of additional hardware resources coincides with the timing and scope of the requirements. Form 1 in the attached forms appendix is a suggested questionnaire that can be used to solicit input from the applications programming group to the operations group.

Production, of course, is the name of the game both in manufacturing and computer operations. Traditionally, R&D designs a product which manufacturing engineers translate to production specs in a prescribed standard format. Likewise, business systems analysts, responding to user needs, develops a system design. That design is submitted to systems and applications programmers who develop the system, including batch jobs, according to department standards. When complete, operations scheduling or a separate quality assurance group (sometimes a maintenance programming group) reviews and accepts the system and supporting documentation.

In the manufacturing environment, production control accepts engineering designs and develops assembly plans and production specs. Materials and manpower requirements are identified and scheduled. Production then assembles the product according to specs. The process is gradually refined and statistical data is collected to provide feedback to R&D and production control.

Likewise, operations schedules batch processing requirements which will determine staffing levels and needed resources such as run time, disc space, paper, special forms and tapes. As jobs are processed, operations monitors console pages and logfiles to compile statistics on run times, frequency and cause of job failures. These statistics are forwarded to applications for resolution of processing problems.

Conversely, the results of the processing cycle are reviewed by data control personnel who provide statistical feedback to operators on missed schedules. These personnel typically distribution staff, tape librarians or I/O control staff provide a system of checks and balances that insures adherence to processing procedures and documentation, as well as providing continuous validation of those procedures and documentation. Schedule volumes or concurrent job mix is adjusted to available processing time. Similarly, the manufacturing Q.A. function reviews production, providing feedback used to improve both the product and the manufacturing process.

Finally, the product is packaged and shipped to the customer. Defective or damaged product is returned and repaired, and repair cause and frequency statistics are kept to improve future product reliability. Computer operations distributes batch output to the users. If inaccuracies exist, either in the form of program bugs which cause job aborts or unsound data, applications makes needed modifications. Feedback from operators can also be used to optimize future design.

Understanding the composition and function of the model is only the first step. Optimizing it is the critical second step. Perhaps the single most important factor in successfully controlling computer operations is the establishment and enforcement of standards. This has long been recognized in manufacturing where standard "assembly line" techniques have been used for over 150 years. Beyond providing a continuity, adequate standards can minimize the need for operator training while maximizing flexibility. The three major candidates for standardization; jobs, documentation and schedules.

A caution; to effectively develop and maintain standards will require the cooperation volitional or otherwise, of the applications and operations groups. That may prove difficult. Programmers, being the creative, free spirited, willfull, overpaid group that they are, often resent the imposition of standards. Management may have to flex managerial muscle to ensure a happy compliance.

Jobs names can be standardized to identify production jobs, updates, reports and the specific applications group to which the job belongs. It can be as generic or specific as site requirements dictate as long as the naming convention is observed by all. Keeping the convention simplistic is also esential since a complex naming standard can easily degenerate into no standard at all.

Input and output priorities should be standardized to allow critical jobs first consideration when the job queue is full and there are several jobs waiting to be processed. The more critical the job, the higher the inpri. Outpri can be used to identify forms type. An outpri of 9, for example, can indicate single part paper, while an outpri of 12 may indicate three part paper, and so on. This will allow the operator to print all available output of a certain type before changing paper, rather than constantly changing paper to accomodate the

next available spoolfile. Anyone who has ever received four additional copies of a report they did not need, knows that relying on operator memory is risky at best. Enforcing use of the forms message capability for all special forms also insures that purchase orders do not print on sales order forms.

Tellop formats for tape mounts or other operator interface should be visually imposing to instantly alert the operator. That failing, a few well placed control G's can disturb even the most determined grave shift slumber. Tape names should have some meaning. Avoid generics like "DBSTORE" and "T". If 20 tapes are all labelled DBSTORE even the best operator is playing Russian Roulette trying to correctly mount and identify each one.

Standardize both the format and content of the tellops and insure that all information necessary for the operator to adequately label all tapes is presented on the console. (You to of course use preprinted tape labels either of the commercially available sort or stock blank labels printed with your very own 10 line BASIC program.) Tape labelling data sent to the console should include the name of the job creating the tape, the tape name (as identified in the formal designator defining the file), the retention period, the usage (should the write ring be in or out?), the BPI and a brief description of the contents (keep it short, tape labels are little tiny things!). Other useful information includes the date, the operator, the system (for a multimachine environment) and the MPE release. (Ever try to read a 2 year old store tape with a different blocking factor?)

Standardizing output file names helps too. It is difficult to sort out twenty reports with the useful title of "LP". A few well placed file equations can work miracles. Use comments liberally. Functions and restart points should be clearly commented. Job standardization can be a tremendous help to operations. Take full advantage of it.

Documentation standardization is equally helpful. It simply means that format should be the same for all jobs in allsystems. One of the problems with documentation is that many people do not know how to use it. If it takes more than a few seconds to locate the desired information, they either ask someone for the answer or simply give up. Getting accustomed to using one style of documentation creates a comfort level when using new and old documentation alike. People understand how the data is organized, and what information they can expect to find.

Documentation should include diagrams to illustrate job flow. Restart and recovery instruction should be clearly defined for each job, and those jobs which can not be restarted without programmer intervention should be identified. Detailed documentation for each job should include: the system on which it should be run (if multiple systems are used), the name of the programmer responsible for maintaining the job, the jobname.group.account where the production version is located, the job function, i.e. report, update etc., a short description of what the job does, any tape, microfiche or page print requirements, estimated run time, output and sector requirements, whether the job is critical or not, and whether the output is confidential, the schedule it runs on and any jobs or events that must preceed it.

One technique for standardizing documentation is to use forms. Forms insure that information is presented in a standard format, that only necessary data is provided and, conversely that no essential data is omitted. Use of forms creates easily updated documentation since operator manuals can be kept current by simply substituting a current form for an out of date one. Forms also provide excellent input documents for automated documentations and scheduling system. (See form samples in Appendix.) An automated system, whether homegrown or purchased provides strong benefits for the operations group struggling to keep track of hundreds of jobs. It should be considered essential in a multimachine shop.

The last question, once a standard documentation process is created, how do you maintain it? How do you quarantee that the programmer caught up in the thrill of the creative process comes to earth long enough to tell operations how to showcase his masterpiece to best advantage? Stockades, whipping posts, and public executions are one method. A protected library is a more humane alternative. The library is a group or an account to which only operations has write and save access. Read and/or execute access may be globally available. All production jobstreams are transferred to this account by a designated individual or group. The scheduler, I/O control or Quality Assurance groups are natural choices. Operators execute jobs from this library only and jobs are transferred to the library only after receipt of the appropriate documentation. This provides an easy, fairly administered, system to "remind" all concerned to keep documentation up to date. The only secret to maintaining such a library is enforcing a "no exceptions " policy. Any job requiring any form of operator attention--be it to stream it, alter its priority, mount a tape or print the output should come from the library. Take note; systems programmers and technical support personnel tend to try to "beat the system" more than applications programmers. A few violators can make the whole exercise meaningless so don't allow exceptions.

With such detailed documentation of each job, the operator should be able to handle any contingency, from rescheduling production due

to prolonged system failure, to adjusting copy counts. Of course not all jobs run each evening. Standardizing schedules will alleviate the last minute scramble to identify which jobs genuinely qualify for production.

Scheduling: the art of detetermining which jobs run on which days and what resources (disc space, tapes, paper, special forms, DS line availability) are required. A good scheduler is a pearl beyond price; part wizard, part puzzle master, a mythological hero who can unravel the complexities of the Gordian knot before breakfast (or score millions of points on Space Invaders) and thinks nothing of planning 500 jobs running 8 at a time on each of 3 systems. Accurate scheduling for other than the smallest operations is impossible without automated assistance. A good scheduling system is linked to a strong documentation system and allows planning on a daily, weekly, monthly, quarterly and annual basis. However even an MPE file maintained in the Editor and printed with a simple report program can provide adequate automation. Schedules should be printed in a standard format for the operators and should include references to any necessary documentation. A section for operator comments and feedback helps answer the all important question "what ran last night?" Log file extract programs can provide an automated feedback process for the scheduler providing run time statistics, average lines of output and other essential detail down to the process level if needed. Again the contributed library is a good source for programs to read and report log files. Experiment with available programs and see what meets your needs. Remember too that operations is a dynamic environment. The program that does exactly what you need today may be extraneous overhead six months from now when your needs change. Review and refine the statistical process just as statisticians do for a production line. Use the logfiles to help focus on your current needs not just to produce reports that fill file cabinets and aren't used.

A final word about standards; the best standards are simple, are written down for general reference, and are enforced by automated means. A computer is impartial and consistent; a computer is never uneasy or embarrassed when asked to serve as a review or enforcement mechanism for coworkers; and a computer can't be bribed to make an exception "just this once."

A second manufacturing technique that can be used to optimize the operations environment is the elimination of repetitive manual labor. On the assembly line this means using everything from power tools to robots to eliminate the necessity for human intervention. This reduces manpower requirements and minimizes the human error factor. For the operations environment this means eliminating operator involvement in streaming jobs, manipulating spoolfiles or editing job files. Standards contribute to this, particularly a standardized outpri/ outclass/ jobfence scheme that reduces spoolfile manipulation.

Another technique is to coordinate creation of "master" jobs with the scheduling function. These are jobs with no function other than initiating sets of jobs according to the prescribed schedule for that evening. Use of JCW checking to trap program and job aborts and to control sequence requirements can significantly reduce operator involvement in job processing and control. Let each job start its successors based on conditions trapped by JCW. These conditions can also contribute to maximizing throughput by reducing system "dead" time, that is the time lost waiting for the operator to initiate the next step in a sequence of jobs. The contributed library is a good source for programs to help speed throughput and reduce wasted "dead" time. Remembering that the HP3000 is a multiprocessing system is key to maximizing throughput. A Series III can process 5-7 jobs simultaneously; a Series 64 10-12 jobs. It may be necessary to experiment in your environment to find the optimum job mix. To do this it is first essential to create jobs that can run simultaneously. Avoid such limiting processing techniques as exclusive file or data base access, clustering simultaneous events (several reports for instance) in one job rather than many.

In the same vein use jobstreams and UDC's wherever possible to streamline routine operator functions. These can include UDC's to set system paramenters (like jobfence, outfence, jobpri and limit) at different points during the day or to control devices (open DS lines, spool printers) after system restarts. Jobstreams can be used to perform routine diagnostic and maintenance functions like running MEMLOGAN and FREE2 daily or performing a weekly disc condense. Techniques like these that create a user friendly environment are a routine part of the implementation of any application system. Operations however is like the proverbial cobblers children who have no shoes, is generally the last place where normal good data processing practices are applied. Review your operations environment. Look at it as a connected system. If all the segments are preforming to your expectations guard your operations manager well; if not consider whether the solution lies in changing the approach not the people.

How much improvement can be expected from implementing these techniques? Well, like everything in the HP3000 world, that depends on a number of factors. Things like how big is your installation, how complex are the applications and how unstructured is the environment when you start? In a multiple machine environment (3 or more systems) clear

definition of the scheduling and support functions and implementation of standards can produce throughput improvements of 30% or more. More significant is the reduction in the "hassle" factor. This is the level of stress created when programmers are called in the middle of the night, when users complain constantly about systems constantly down or output that is late or incmplete everyone's life (especially the operations manager's) is very stressed. Reducing this element decrease turnover increases communications and teamwork and creates a professional data processing environment. This is an immeasurable benefit; the most significant one that can be achieved!

APPENDIX

Form 1 - Capacity Input

Form 2 - Tape Label

Form 3 - Job Documentation

Form 4 - Operations Procedure

Form 5 - Distribution

Form 6 - Special Forms Print Documentation

Form 7 - Production Turnover (Scheduler's Instructions)

Form 10 - DataBase Capacity Change (Adager Request)

Biographical Sketch

Victor Rozek

Educated at San Francisco State University, Mr. Rozek has been involved in computer operations for the past eight years. He has held management positions with SILTEC Coporation, a California based silicon manufacturer, and Smith-Kline Clinical Labs, He is presently Data Processing Operations Manager for ITT Qume Corporation in San Jose, California.

Sheila Dummer

Ms. Dummer received a B.S. from the University of San Francisco and continued with graduate studies in the MBA program at that university. She has worked in data processing operations for 15 years holding management positions in computer operations, systems programming and applications system development with Wells Fargo Bank and ITT Qume Corporation. She has consulted in the area of operations optimization for the HP3000 for 5 years and has worked with the HP3000 since 1973. Ms. Dummer is currently Computer Operations Manager for the CBX division of ROLM Corporation.

SALES FORCE
R&D ENGINEERING
FIELD REPAIR
CUSTOMER
BUSINESS SYSTEMS ANALYSTS
BUSINESS SYSTEMS PROGRAMMERS (DEVELOPMENT)
MAINTENANCE PROGRAMMERS
USER
USER NEEDS
FORECASTS REQUIREMENTS
PRODUCTION SPECIFICATIONS
PRODUCT IMPROVEMENT FEEDBACK
PROBLEM FEEDBACK/CORRECTION
DEFECTIVE PRODUCT REPAIRED
PRODUCT SENT TO CUSTOMER
TECHNICAL SUPPORT (SYSTEMS PROGRAMMER)
SCHEDULER
OPERATORS (PRODUCTION)
I/O CONTROL TAPE LIBRARIANS
DISTRIBUTION
PRODUCTION PLANNING/ MATERIALS
PRODUCTION CONTROL
PLANS, MATERIAL, EQUIPMENT, ETC.
ASSEMBLY
PRODUCTION VERIFIED
Q/A
PRODUCT PACKAGED/SHIPPED
SHIPPING

IMAGE/3000 - The Cost of Single Threading in a Large Data Base.

William M. Lyons
Senior Technical Analyst
GTE Data Services, Inc.

HP - Higher Performance? Yes. HP continues to produce larger and faster computers. However, as with anything man-made, weaknesses can and do exist. When developing or even maintaining applications on the HP 3000, it is vital to keep these weaknesses in mind or the applications become vulnerable through these points. For the HP 3000, one of its strongest points even has a weak point. IMAGE/3000 was developed in a time of slower CPUs and slower disc drives with only one controller. Even the maximum number of terminals that could be connected to the system was much lower and the baud rate for terminals was 2400 not 9600 or 19200, now possible with the HP150 and ATPs on a Series 64. This paper is not intended to fault IMAGE/3000, it is just to make all users aware that there are problems that can be overcome in our applications, since HP has stated, previously, it would require the rewriting of IMAGE to correct them.

Today, I will discuss our reasons for being concerned with performance, our findings through our investigation, and the solutions we have implemented or considering to implement. Those involved in the testing included Dorann Barenbrugge of GTE, Hugh McKee and Mark Conroy both of the HP Tampa office.

I work in the Information Systems Group where we develop application systems to meet the needs of the telephone operations area of GTE. Our applications are developed to serve the common needs of all the various telephone companies in GTE. Some companies are very large while others are small. They differ from company to company on needs and we have to develop a core system with flexiblity. The specific project I work on is called CNAS, Circuit Network Administration System. About ten years ago, this area was mechanized using an IBM mainframe with sequential master files that were updated once a week. This system recorded the makeup of circuits which connect the various central offices of the phone company. It also recorded the makeup of Special Service Circuits (leased lines, conditioned lines, extended local area service, etc.). Some of the transactions for this system were quite large due to the amount of data required to describe a circuit or its facilities. The result was a lot of grief when a transaction of 24 cards long had 1 character punched wrong. To eliminate many of the simple errors, a front end data collection system was developed on the HP 2000. Weekly, the transactions were extracted to tape and processed through the IBM batch system.

The overall combination of the IBM and HP 2000 made the data smoother to enter because the HP 2000 pointed out simple edit errors on input. However, it did not make the data accessible on demand. Therefore, we were authorized to develop an online system to replace the original IBM/HP 2000 system. After an investigation into several CPUs, the HP 3000 was chosen as the host system. The development took about two years and was installed in the first telephone company on a Series III with seven 7925 disc drives, one of which was a serial disc drive for backup purposes. More terminals were connected than originally estimated. As other departments found out that they could retrieve data useful to them, they requested access and a terminal. Also, the users found that data not originally anticipated could be stored in the data base. This added up to more than the system could handle. More disc space was added and more memory, too.

The Series III was monitored by our Hardware/Software Evaluations group and found to have a bottleneck on CPU power. At the same time HP was introducing the Series 44. One was ordered to replace the Series III and arrived about one year after the application system went live. In the same time frame, MPE III was being replaced by MPE IV. The combination provided the extra power needed. With the Series III, 21 - 22 concurrent sessions was the maximum before response times would deteriorate. With the new Series 44, no problems were noted until more terminals were added several months later. The limit now seemed to be about 35-37 sessions.

About this time, we became more involved with the performance investigation. The first thing we did was to monitor the production system with OPT/3000. We did this using both batch and online observations. The batch mode was set up to report every hour and print once a day. We found the system to be disc I/O bound. The CPU was busy 52% of the time while it waited on disc I/O 27% of the time. The disc I/O rate was only 34.6 I/O's per second. With two controllers, this seemed low. This prompted us toward looking at why and eventually led us to the Global Data Base Control Block.

Each user has a local control block for the data base for each DBOPEN but only one global control block exists for a data base and only one user can perform an intrinsic call against the data base at a time. This concept is called Single Threading. For example suppose three users are accessing the same data base. User A wants to do a DBPUT, User B wants to do a DBUPDATE while User C wants to do a DBPUT also. They would queue up as follows:

1. A would take ownership of the global data base control block.
2. All the disc I/O and other work required to place the record would be executed prior to releasing the global control block.
3. B would be able to take ownership and execute its DBUPDATE releasing the global control block after it is finished.
4. Finally C would get the global control block and execute its DBPUT.

Every user must funnel through this control block one at a time to execute an IMAGE intrinsic call against the same data base.

Among other observations, we saw that most of the users were using the data base for online inquiry or producing printed reports in a Query job. This proved to provide a solution later. However, the new Series 64 was being introduced. We did some testing in the Atlanta HP Sales Office to first insure compatiblity with the application programs. (We were in development before VIEW came out and developed our own terminal I/O routines.) We also checked the performance of the Series 64 and to obtain a comparison, tests were repeated on the Series 44 and Series III systems in Tampa. All three systems were similar. The Series 64 had two 7933 disc drives. The Series 44 had ten 7925 disc drives. Both of these systems had two masters while the 44 had additional slave drives. The Series III had one master 7925 disc drive and three 7925 slave disc drives. The Series III had look ahead seeks enabled. The test showed that there is an increase in capacity, until IMAGE, with one data base, was introduced.

The following measurements support these findings. The measurements are split into two parts. Part I deals without IMAGE. Part II deals with IMAGE.

I. Performance using an evaluation program.

The program was written by the Hardware/Software Evaluation group of GTE Service Corp. It allows the user to vary the load forced upon the system by writing to disc, tape, printer and punch card devices or CPU usage. For the purposes of our evaluation, five measurements were taken. Disc I/O rates were those observed using OPT/3000.

A. CPU only -

Job with 450,000 loops through a series of arithmetic equations.

	III	44	64
Elapsed time (sec.)	1087	687	233
CPU time (sec.)	1038	675	231

This is the strong point of the Series 64, being about three times faster than the Series 44 and about five times faster than the Series III.

B. Disc I/O - One drive only

Job with 10,000 physical writes to the system disc.

	III	44	64
Elapsed time (sec.)	246	245	239
CPU time (sec.)	106	72	40
Disc I/O rate per second	40.5	41.4	41.3

Again, the CPU is faster, but, the elapsed time barely dropped from the Series III and Series 44. However, this was using only one disc drive and one controller.

C. CPU and Disc I/O together -

Job with 10,000 disc writes using a single disc drive and 100,000 loops through the same arithmetic equations.

	III	44	64
Elapsed time (sec.)	594	400	236
CPU time (sec.)	543	361	142
Disc I/O rate per second	17.1	25.2	42.9

This points out that because of the faster CPU, a mixture with CPU and disc will spend less time on the CPU dependent activities and be allowed to process more disc I/O.

D. Disc I/O and CPU-Heavy

Two jobs running concurently that were used for parts A and B.

	III		44		64	
	A	B	A	B	A	B
Elapsed time (sec.)	1152	1259	713	850	239	432
CPU time (sec.)	1037	105	676	71	232	40
Disc I/O rate per second	4.0/41.0		3.7/41.4		3.8/42.1	

The Disc I/O rate shows as the lower figure when both jobs were competing on the system. The higher figure shows the rate that the disc I/O job achieved after the CPU bound job completed. Again, this points out a very fast CPU but disc I/O to a single disc drive about the same for the Series 64.

E. Disc I/O - 2 drives

Two jobs running concurrently that were the same as in part B but going to two different drives. On the Series 44 and Series 64, the drives chosen were on different controllers. Job A below is for writing to the system disc; Job B is for writing to another drive.

	III		44		64	
	A	B	A	B	A	B
Elapsed time (sec.)	402	394	252	473	266	239
CPU time (sec.)	99	99	73	70	40	40
Disc I/O rate per second	49.0		62.5		84.7	

This shows the improvement in disc I/O rates that could be achieved in the upgrading of the hardware. In fact, our further testing on another series 64 with four 7933 disc drives and four jobs writing concurent to separate disc drives pushed the disc I/O rate to 116.6 per second.

II. Performance through the application programs using IMAGE/3000

Another phase of testing on the HP 3000 Series 64 and Series 44 depict the way the application will work. This was accomplished by eliminating operator think time through setting up the online programs to run in batch mode. To do this, the terminal I/O routines were subsituted with routines to read input from files on disc. The modification was only a small amount of code compared to the total program in each case. Within the read routine, a time stamp was established to give us response times for the various transactions. The system was again observed, during these runs, using OPT/3000.

Several measurements were taken.

A. Each online mainline was run, one job per mainline, for a total of eight jobs. Activity was against the system disc only. Listed below are OPT/3000 observations:

	44	64
CPU Busy	24%	15%
CPU Paused for Disc	72%	82%
Disc I/O Rate Per Second	29.1	30.7

Transaction	Response time/CPU time in seconds	
Simple type 1	18.1/00.279	28.3/00.122
Simple type 2	6.6/00.114	2.9/00.060
Heavy I/O type 1	433.2/22.455	479.1/12.083
Heavy I/O type 2	603.7/81.308	669.6/47.571
Medium type 1	24.5/00.761	39.6/00.369

Total by job	Elapsed/CPU time in seconds	
1.	153.1/003.821	203.3/01.818
2.	1585.9/182.214	1686.7/98.060
3.	1449.9/178.620	1466.9/98.616
4.	319.8/012.663	352.9/06.471
5.	142.8/004.620	187.2/02.069
6.	31.4/001.405	63.6/00.636
7.	134.4/018.261	332.5/09.270
8.	161.7/005.410	141.0/01.926

The above results show again the much faster CPU of the Series 64, but elapsed times not any better than the Series 44. The figures show worse times for the Series 64 in some instances. There may have been a bias to the test as job 6 dropped out prematurely for an unknown reason. The overall effect shows that the Series 64 with IMAGE could get about 30.7 disc I/O's per second as opposed to the Series 44's 29.1 I/O's per second.

B. The same test was done again but now the Master and Detail data sets were split across disc drives and controllers.

	44	64
CPU Busy	24%	15%
CPU Paused for Disc	67%	78%
Disc I/O Rate Per Second	29.2	35.9

Transaction	Response time/CPU time in seconds	
Simple type 1	25.8/00.270	17.4/00.118
Simple type 2	5.2/00.120	5.9/00.052
Heavy I/O type 1	495.7/23.048	396.7/12.526
Heavy I/O type 2	761.7/81.073	630.2/47.463
Medium type 1	33.3/00.732	28.5/00.397

Total by job	Elapsed/CPU time in seconds	
1.	181.7/003.800	149.6/01.794
2.	1928.9/182.586	1528.8/98.210
3.	1756.1/178.674	1400.8/98.475
4.	347.7/012.870	282.3/06.637
5.	166.4/004.640	140.9/02.247
6.	107.1/005.370	102.3/02.574
7.	152.5/018.307	121.9/09.371
8.	190.8/005.535	161.6/02.784

This configuration, with two disc drives, is closer to the model of a typical site for our application. The results show that with a Series 64, improvement can be expected. However, this improvement is small compared to what had been hoped with upgrading to a Series 64.

C. This test consisted of two parts. The first had eight jobs adding and deleting records through the programs in such a way as to cause heavy IMAGE activity. All activity was against a single data base. The observations were as follows:

	44	64
CPU Busy	25%	15%
CPU Paused for Disc	70%	83%
Disc I/O Rate Per Second	29.9	30.9

The second part was the same as the first part except four data bases were used instead of one. There were two jobs per data base. The observations were as follows:

	44	64
CPU Busy	34%	20%
CPU Paused for Disc	55%	73%
Disc I/O Rate Per Second	44.6	58.3

This shows that the systems are capable of doing more disc I/O. Each data base has a global control block which allows only one IMAGE Intrinsic to be satisfied at a time on a system level. With four data bases, there were four global control blocks, and four intrinsics calls were being satisfied at the same time. For the Series 64, this meant almost a 100% increase in the disc I/O rate.

D. To further confirm the bottleneck with the Global Data Base Control Block, this test was conducted with two jobs in part A. Both were heavy on the IMAGE calls for disc I/O. They were accessing totally independent data sets in the data base. The first part again used only one data base.

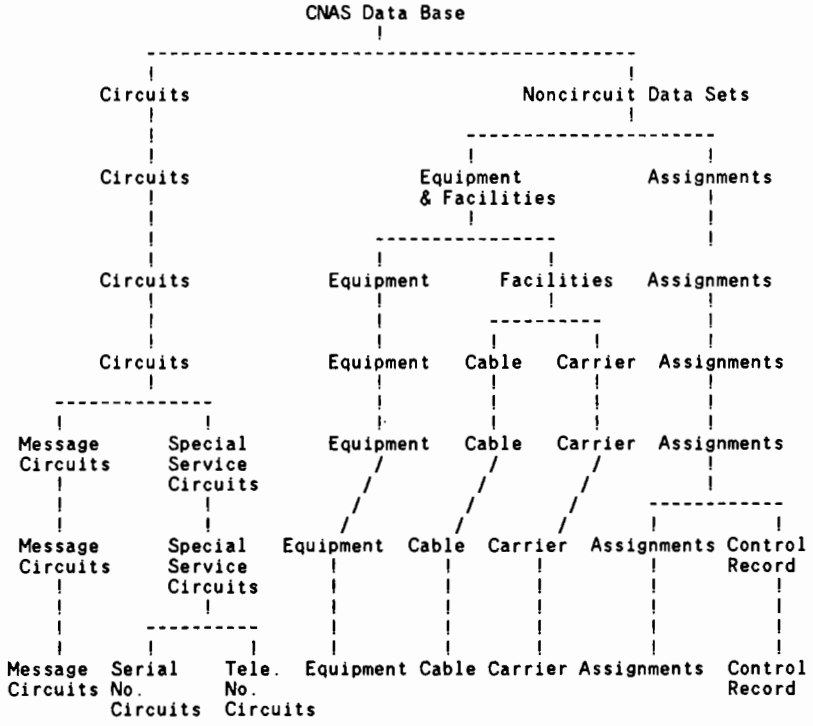
	44	64
CPU Busy	36%	13%
CPU Paused for Disc	58%	85%
Disc I/O Rate Per Second	26.3	30.0

Submitting the same jobs against separate data bases gave the following results:

	44	64
CPU Busy	35%	18%
CPU Paused for Disc	58%	78%
Disc I/O Rate Per Second	38.6	44.0

Again, this shows an increase in the disc I/O rate.

With this information, it became clear that more than an upgrade to the CPU would be required. We developed programs to simulate the transactions of the application, which would also allow the data sets to be moved from data base to data base as needed for the splitting. We also took our original data base schema and developed a logical breakdown of the data sets first into two data bases, then three, four, five, six, seven, and eight data bases. In some cases, an automatic master was duplicated in two separate data bases in order to place two detail data sets pointed to by the automatic master in separate data bases. Each level below is a depiction of the breakdown.



The following is a synopsis of the various schemas and the change in space requirements.

Number of Data Bases	Number of Data Sets (Root Files/ other)	Number of Paths	Total Disc Sectors	% Increase from One Data Base
1	1 / 78	138	300,587	-----
2	2 / 78	138	300,595	.0266
3	3 / 79	138	297,546	-1.0116
4	4 / 80	138	302,441	.6167
5	5 / 81	138	306,830	2.0769
6	6 / 90	146	305,784	1.7289

7	7 / 90	146	305,790	1.7309
8	8 / 104	164	304,659	1.3546

In a few cases we also split detail data sets that were shared between categories. The capacities were adjusted to handle the same amount of data and as the one column indicates the space required stayed about the same.

Our testing was done on a Series 64 with four 7933 disc drives. The test was conducted but the results were somewhat unclear. Therefore the tests were repeated with more tools on two separate occasions. Our local SE's were involved in these tests and the two repeat sessions.

The first phase of testing was observed using OPT/3000 with 16 jobs running in the background. It was run and set to produce summary reports for each minute of run time. In all cases, the reports showed an initial load time that involved loading of the programs, opening the data bases, and other activity. Eventually, the figures became fairly stable. The following table represents an average of the stable figures:

Number of Data Bases	CPU %Busy	CPU %Paused Disc	Disc I/O Rate (per Second)
1	19.4	77.5	24.8
2	30.3	61.8	44.0
3	31.0	60.7	44.0
4	47.0	39.5	56.0
5	72.0	11.8	45.6
6	60.5	22.8	52.3
7	62.2	22.2	49.6
8	64.3	19.3	54.7
*0	63.3	11.3	116.6

* This measurement was taken through a program not using IMAGE to establish the maximum capacity of the system involved. Four jobs were run concurrently with each writing to a different disc drive.

It was anticipated that, as the system disc I/O rate increased, the CPU would become busier and that there would be a smooth curve produced by the figures. However, there was an unexpected sharp jump in CPU busy and drop in the disc I/O rate for five data bases as opposed to four data bases. The test was repeated that same day to confirm that no unknown influence had biased the test. The same results were displayed in the reports from OPT/3000. Our local SE's, who assisted in much of this endeavor, met with us to review the results but were not able to determine any reason for the drop. We again repeated the test, so they could monitor it with APS/3000 (Sampler). It was hoped that we could pinpoint in what code the jobs were spending more time with five data bases. The test was conducted with the same program that was used before. This time, to give a contrast, the test was conducted with four and five data bases with twenty and eight jobs running. The following are the results of that test:

Number of Data Bases	Number of Jobs	CPU %Busy	CPU %Paused	Disc I/O Rate (per second)
4	8	68.0	15.0	53.4
4	20	56.0	27.2	56.8

5	8	76.0	10.0	39.8
5	20	74.4	6.8	51.4

The monitoring with Sampler provided very little insight. It showed no place in the code that stood out as taking most of the CPU time. The extra CPU time that was required by five data bases was evenly spread between all code segments. There was a slight increase in one of about 4%. All of the other segments were less.

After further consulting among the HP software personnel, locally and elsewhere, the test was repeated again. This time, the system was monitored by OPT/3000, APS/3000, and MPE Data Collection. The time of observation was extended from ten minutes to thirty minutes. Also, thirty-two jobs were used with one, four, five, and eight data bases to draw a comparison. The results were as follows from OPT/3000.

Number of Data Bases	CPU %Busy	CPU %Paused Disc	Disc I/O Rate (per Second)
1	24.0	71.0	26.8
4	58.3	20.8	56.1
5	73.8	4.0	52.8
8	66.0	8.0	64.0

The results were again inconclusive as to why there is a change between four and five data bases. Sampler and MPE Data Collection did not show anything. The files of data collected were passed onto senior technical people in HP, but, nothing has been resolved, to date. Therefore, it was concluded that the change we would make had to be flexible.

We learned how to manipulate the data base into multiple data bases.

1. Using Query, produce a "Form Sets" listing. Number the data sets in sequence. The root file name plus this number gives the MPE file name for that data set.
2. Draw up new schemas using the capacities shown in the "Form Sets."
3. Store the data base to tape and then purge the data base using DBUTIL.
4. Create the root files by using DBSCHEMA and the new schemas.
5. Create the data bases using DBUTIL and the root files. This is required because the root file has an indicator that shows whether a data base has been created or not.
6. Using Query, produce a "Form Sets" listing for each data base.
7. Using FLUTIL3, change the root files from PRIV to NONPRIV files. (File code for the root files are -400 and other data base files are -401. The file code can be changed to 0.)
8. Using DBUTIL, purge the new data bases. The root files not marked PRIV do not get purged.
9. Using FLUTIL3, return the root files to PRIV.
10. Restore from tape the data base files, except for the root file.
11. Using FLUTIL3, change these files from PRIV to NONPRIV.
12. Using the new data bases' "Form Sets" produced in step 6, draw up the MPE file name for the data sets, using the root file name and sequence of the data sets.
13. Rename the data sets as required to move the data sets between the old data base and the new data bases.
14. Change all the data sets back to PRIV files, using FLUTIL3.
15. Recheck your steps. If possible do this on a printing terminal or one with an integral printer and log everything. Also do it with a few people double checking your work by watching you

over the shoulder. It is very open to problems and should be done with the utmost of care.

We concluded from our work, that it is better not to split paths and create duplicates on any data sets including automatic masters. However it can be done. Before storing the data base to tape, remove paths that will be split. Create the schemas without these paths. Then, readd the paths, after step 14. We found that using Adager to delete the path and readd it was fine for our test data bases. However, some of the production data bases which have about 3,000,000 sectors involved make it difficult. Also, to make it flexible to meet the needs of the various companies we couldn't duplicate the name of of a data set.

The solutions are as follows:

1. The Series 44 was kept as a backup system for the application and noncritical additional applications. The data base is stored to tape every night from the main system, a Series 64 and restored to the Series 44. The Series 44 is used for listings only, by those who can use the data from the previous day rather than the current data base. It also is used for query since the data is never more than 24 hours old and most of the inquiries did not need any more current data. If the up-to-date data is required they can shift to the Series 64 and repeat the inquiry.

   This same procedure could be done on the same system using GETFILE2 to restore the data base into a separate group or account if needed. The result is two global control blocks for the two data bases which will improve the retrieval time. However, twice the disc space is required.

2. Split the data sets between data bases and thus allow more con- current activity. If you are just designing the application system, this will be fairly easy. Keep it flexible because activity will probably vary from what is anticipated. If you have a system already developed you may need to alter your programs. Our plan is as follows:

   A. Divide the data base as above without splitting paths. In other words any data sets that are linked by a path or series of paths will not be split into different data bases. B. Alter all the programs to call an indexing subroutine. This routine would be passed the data set name and would pass back the data base name to be used in the subsequent IMAGE intrinsic call. This routine would also issue a DBBEGIN when the intrinsic call is the first for a given data base and the call is for DBPUT, DBDELETE, or DBUPDATE. This routine should be very efficient in looking up the data base name. In our case each data set has a numeric name and thus could serve as an index into a table. C. The calls to DBBEGIN are removed from the programs since the indexing subroutine will take care of this. Where DBEND is called now, a subroutine would be called to check a table of the data bases and issue DBENDs for all the data bases that a DBBEGIN was issued against. This is called stacked DBENDS. This limits the window that a transaction could be marked complete for one data base and not for the other data bases involved. Unfortunately IMAGE does not allow transactions to be marked as covering multiple data bases. A program to analyze the log file could be written to change DBENDs to abnormal DBENDs. This may not be needed because the window for problems is so very small. Only a system failure in the middle of a stack of DBENDS would cause a problem. D. We planned to monitor the effect of splitting 4 and 8 ways with the production data base at a phone company. The routine in part B would be issued to all phone companies with the option of having 1, 4, or 8 data bases involved. E. There would have to be tighter controls over backups to keep all data bases in sync. Recoveries from log files and restores from backup would also need very tight controls. Perhaps HP can assist us by providing a way to define a transaction accross multiple data bases and have DBSTORE/DBRESTOR handle multiple data bases on the same set of tapes. This would eliminate a great deal of the uncertainty of multiple data bases.

   These are still only plans. Due primarily to the promises of HP in the area of Disc Caching approval to go ahead with this change has not been given. The change for us is quite costly due to the size of the application. The decision is to wait until measurements can be taken from Disc Caching. However, Disc Caching may not provide the performance improvement needed to large data bases and it is anticipated we will eventually be given approval.

We could not find instances of simple data entry in our application. However, it is possible that the use of message files that would be added to online and read by a batch program

could speed up the entry work. The batch program could do the actual DBPUT. We do not have figures from testing this but it has been suggested as a way to improve performance and it seems logical. However, spliting the data base is the most realistic way to improve performance.

Splitting the data base has some other advantages. First, IMAGE has maximums for the number of data sets, items, etc. that a single data base may have. By splitting the data base there is more room for growth before these maximums are reached. Second, as it stands now, the data base is too large to go through a normal DBUNLOAD and DBLOAD to correct broken chains or reorganize the data base. It was done one time and took about a week to complete the whole process. With smaller data bases, individual data bases could be done on weekends. As it stands now, to correct broken chains a person has to do it by hand using DBDRIVER.

To sum this up in a very few words, IMAGE/3000 is better used as a handler of multiple data bases than of one giant data base. Large applications should not depend on IMAGE to keep the data in separate data sets in sync by keeping the data sets in the same data base. When this happens the response time will increase and the disc I/O rate will decrease.

In essence, do not keep all your eggs (data sets) in one basket (data base).

William M. Lyons is a Senior Technical Analyst for GTE Data Services, Inc. in Tampa, Florida. He has worked on applica- tions on the HP 3000 for over six years. A native Floridian he graduated from Florida Atlantic University in Boca Raton with a degree in Mathematics, and a minor in Computer Science. He also holds a Masters degree in Mathematics from the same institution. Currently, Bill is serving as the Vice Chairman of the HP Florida Regional Users Group, FLORUG.

Networks - It's a Jungle Out There
Table of Contents

V. A. Shoemaker
November 1983

I. Problem/Situation

This is not the definitive technical paper of the year on designing, implementing, and supporting a nationwide telecommunications network. My expertise lies not in the strange world of telecommunications, but in the field of business system's design. As part of the project team assembled to build an "information pipeline" to our satellite offices, I got involved in the building, care and feeding of a telecommunications network which was started in January of 1981.

Our company had a problem or opportunity, unlike most organizations that are not totally centrally located - the need to get information to their satellite offices in a timely manner. The method that our company used in the satellite offices to get information back and forth was a DEX machine.

This device allows the user to transmit paper images via telephone lines to another office that has a similar device. The fastest these devices can transmit is one page (8 1/2 x 11) in three minutes. The print quality and the reliability were poor. The device, as you can imagine, also has limited appplication. The device only transmits paper images. It does not provide permanent storage of the data nor does it have any ability to analyze it.

It was decided that we were to establish an "information pipeline" to our satellite offices - a good idea and obviously an idea that required MIS involvement and innovation. At the time of the request, our MIS department had a staff of eight people, none of whom were telecommunications experts. A staff of three people was assigned to the pilot effort of installing a computer, an "application's library", terminals (HP/2624), and printers (slave HP/2631B) out in three satellite offices. The pilot effort had a budget of $30,000 taken from keypunch funds of one of the systems that we had to replace. The $30,000 was to cover application development, three phone lines and telecommunications, lease of an HP3000/30, a MRJE line, 4 terminals, and 4 printers for six months. So you laugh....we were ambitious.

As we were not aware of what options were available to us as far as hardware, telecommnuications options, and software development tools, we contacted the three names in the minicomputer industry, IBM, DEC, and HP. I bet you can guess the ending. The IBM series 34 didn't meet a lot of our needs and was discarded almost immediately. DEC satisfied all of our hardware needs but made little or no attempt to assist us in finding application development tools. We finalized our decision on HP on April 30, 1981 and planned to install our first site on June 1, 1981.

II. Selection of Telecommunication Approach

For our telecommunications needs, being that we were only a pilot effort for six months, we contacted ATT long lines who advised us to use dial-in WATTS (two lines). These lines were to operate at 1200 Baud. The user would call an 800 number and when the high pitch tone was heard, would put the phone on hold and start the logon sequence.

Let me stray a little and tell you of inward bound WATTS. Inward-bound WATTS can be either geographically restricted or nationwide. For our purpose of a pilot to known sites, we selected geographically restricted, which is cheaper. This allows users in a certain geographical area to access our centralized computer via an 800 number.

This approach has some distinct disadvantages. For an example, if you have two phone lines (therefore two geographical areas) and one of the lines goes down, that means that the user is down until the line is repaired. They will not be able to dial-in using the other line. It also means you cannot use the line to dial-in to your computer say for HP remote support. There are ways around these problems, installation of additional lines, etc. but that was not my point. Regardless of these small problems, inward-bound WATTS has some distinct advantages, namely cost. It is especially cheap for low volume (less than two hours a day). It is easy to install as it can run with async protocol, HP's natural mode of operation, and can be connected via an ADCC. You can only go up to 1200 BAUD with ASYNC inward-bound WATTS, so speed can be a disadvantage.

For us, due to our financial constraints, this was the only telecommunicaitons option available to us. Remember that we were only to pilot this effort for six months and could later change our minds with little or no problems. Right? Well, we will get into that later.

III. Setup of the Network

In June, we installed three sites with minor problems. On the system were three applications, all marketing in nature. We had two data entry systems which were developed using V/3000, INSIGHT, and IMAGE. INSIGHT, for those of you who are not aware of it, is a productivity tool which allows you to develop applications with all file structures and V/3000 through an easy-to-use menu driven system. In addition to the data entry systems, we developed a reporting system which allowed the user to request one of three reports based on a limited selection criteria for reports which are resident on our IBM mainframe. I think a description of environment might be helpful at this point. Our company is a classic IBM shop. We have two IBM/3081 and two IBM/3033. Most of our typical applications are large IMS applications. These applications are the source of most of the information needed for our reporting applications.

So for our reporting requests, requests were batched together in some IBM JCL and sent to the IBM mainframe via MRJE, and a disc output file was generated. This output file from the request process was actually a pre-formatted IBM report with IBM carriage control. The COBOL program then read this file in, formatted escape sequences to match the IBM carriage control, and then sent escape sequences to the terminal for it to print the report on the slaved HP/2631B. This whole process is transparent to the user. This process is quite time-consuming and also ties up the terminal while they print their reports, thus restricting them to one function at a time.

We had all of the usual problems with a new installation; the equipment did not arrive until June 1. The HP/3000 was being installed simultaneously with our first training session in the field. Luckily, the installation went with only a few problems. We had developed our applications on MPE IV and were installed with MPE III. We had some initial problems configuring terminals and printers. HP is notorious for unusual cable configurations and cable problems. In our case, the required cables did not arrive with our HP/3000. All in all, the installation went great! We spent three days training the operators, who were secretaries and not familiar with computers at all. They felt comfortable by the time we left.

Our major problem came later in supporting and upgrading a network. Management felt that this was a pilot effort and that at any time one could drop the idea completely by pulling out the equipment or they could switch vendors, or expand the idea to install all our 30 - 50 satellite offices. We found that only looking at the next day or step causes problems when you are managing, installing or planning a nation-wide network.

III. Support and Expansion of the Network

There are three distinct parts to supporting and expanding a nationwide network:

1. The support of the system software and equipment that is installed both at the central site and the satellite offices

2. The support of the applications that are currently available to the users

3. The planning and execution of upgrades to the telecommunications network.

Even though each of these parts are heavily interdependent, I'd like to discuss them separately.

Support of Equipment and System Software

This is one of the tasks that we grossly underestimated. Despite what HP tells you about the ease of use, ease of care, and the wonderful system support that they provide, it is not entirely true. The computer does have certain physical requirements, including a real system manager, not just some Analyst with a System Manager's hat. The job of security, backup and recovery, and system performance and maintenance does not take a couple of hours a week. You need someone who can stay and work through a problem until it is finished and not have to worry about new applications development. When we began the "pilot" we did not have such a person on our staff. This made the problem of hardware/systems software support doublly difficult.

This and the fact that all of our support to the satellite offices was and is handled via phone calls makes the task almost impossible. As you can imagine the inability to see what is wrong is horribly frustrating. I hope that you will never have to experience this. In addition to this we had only inexperienced users. As you can imagine, this makes for some interesting conversations. To illustrate my point, let me give you a typical conversation. Most of our conversations started like this:

User says - My computer is broken.
Analyst thinks - What computer? You only have a terminal and a printer
Analyst says - Can you describe what is not working?
User says - I wanted to request a report and it wouldn't let me
Analyst thinks - Now the computer has supernatural powers. It willed the user's request away.

Analyst says - Start from the beginning and tell me everything you did and everything the computer said
Analyst thinks - Great now I have the computer talking|

Past our number one and number two problems:

1 - No full-time system manager
2 - All support for hardware and system software problems is done over the phone

we had another, which is inherent once you install phone lines into your computer - a two-vendor system. But, you say, they handle two different functions. What problems could ever arise with just installing two simple phone lines. Ah, ye of limited imagination, let me give a simple example that plagued our operation for two months.

Every once in awhile, not often enough to be easily solved, and not far enough apart to be able to be written off as a fluke, our phone lines would hang. We reported this problem to Ma Bell who said - software problem. HP said it was a hardware or phone line problem - noise on the line. We did not know how to troubleshoot line problems. We did not have any fancy line monitors. In fact, when the phone people would ask whether we were using one stop bit or two, we could not answer them. As we were rapidly finding out, to get any service from Ma Bell requires a general knowledge of the terminology, the equipment that you are using, and the configuration of every piece of equipment - computer ports, terminal configurations, and modem strappings. I could venture to say that first-time users like ourselves do not have that knowledge. We were lucky in that HP took pity on us and were able to walk us through that problem.

We now have steps and procedures in place to do troubleshooting of our phone lines. We now know the configuration of each piece of equipment and know the basic terminology. This still has not helped us with the more elusive problems. It would pay, if you plan to install a nationwide network, to send someone to class and then provide them with the appropriate equipment to solve the problems. If you are interested in these kinds of headaches and their answers, Ross Scrugg's TERMINAL TALK in last year's IUG proceedings is a good place to start.

In addition to the support of our telecommunications equipment, we found a great need for all sorts of system software. We have some large databases (80 mg - 100 mg) which, every once in awhile, require additional data items. After spending 16 hours unloading and reloading our data base, we rapidly saw the need for ADAGER.

Applications Support

As you might have been able to predict, most of our support problems were in this area. Not with true application problems, like programs abending or data missing or things just not working like you would expect them to, but with support of the 'how to' questions. You think to yourself, they did train the user. Right? Well yes, but we installed new applications and because of budgetary problems were not able to go back out to the offices to retrain them everytime a new application was developed.

Well, you think, did you send them a user's guide? Yes, we did, but it did not seem to be effective. That still puzzles me. Most times the user would not read the guide, nor would they read the error messages or directions on the screen. They seemed to need that human touch or should I say voice being we were supporting these applications by phone. In addition to the problems listed above, our personnel in the satellite offices changes with some frequency.

In application support we ended up with four major tasks:

1) How do you effectively solve applications related problems quickly and to the user's satisfaction over the phone?
2) How do you introduce and train the user to a new application?
3) How do you handle the problem of a changing user base?
4) How do you minimize the drain that application support has on your analysts' time, as they are still responsible for new application development?

These problems plagued us for a couple of years and in fact are still troubling us to some degree. The solutions that we have come up with fit our problems, but may not solve your situation. We have put the burden of support for applications on the users. We have trained two SUPER-USERS who handle all questions from the satellite offices. We are very lucky to have two users who are at our central site and are bright and energetic. They, like our users in the satellite offices are secretaries/clerks, so they are able to relate on a similar level. For existing applications, they are our first line of defense. They record the problem, answer it if they can (85% of the time they are able to), and then relay the problem to an analyst if they were unable to handle it. For new applications, our SUPER-USERS train an operator at each site over the phone. In addition to this, the user documentation is sent to each of the sites for additional reference. The SUPER-USERS try to train from the user documentation, as it accomplishes two tasks:

1) Makes the user familiar with the documentation
2) Provides an easy-to-follow training guide.

The problem of changing user base is handled one of two ways, either the user is brought to our central site and trained on all the applications by our SUPER-USERS, or is trained by the individual site's SUPER-USER, or if the change in the user base is significant, we send an analyst to their site for complete training of all of their staff.

These solutions have freed up our analyst to develop new applications to add to our "applications library". In addition to this, it has given our user base a consistent and steady contact for problems.

As you can imagine this solution's implementation was not without headaches. Our SUPER-USERS were not initially equipped to handle the 10 - 30 calls a day that we were receiving. Our satellite offices, because of some of the close ties that they had developed with some of our analysts, were not eager to "train" another problem solver. Our analysts, because of the close ties, found it difficult not to get involved. All of these problems took about six months to resolve or subside. I think you honestly need to expect those kinds of problems.

Expanding/Upgrading the network

The end of the first year came around and we were finding ourselves stuck with phones bills worse than the one received in the MCI commercials. Our "pipeline" was a success, too much so. The users were on and using the system an average of four hours a day. We felt this flurry of activity was due to the introduction of ad hoc reporting using QUIZ. This was costing us a great deal and was eating into our funds for application development. Dial-in lines become cost-ineffective when use exceeds two hours a day average. We were way past that point. Although the pilot phase had not officially been cancelled, we needed to explore some data communication options open to us quickly. We explored three:

1) X.25 Packet Switching using Telenet
2) Async/Sync Convertors (MICOM and ATT)
3) MTS/Leased Sync Phone Lines.

I'd like to talk about the technical merits of each and then compare at the end the financial data.

X.25

X.25 is an international standard for public valued added networks. Public data networks allow users of the network to access remote system without having the worry about data integrity. Typically one does not build your own packet switching network but instead buy time and packets from a packet switching network like Telenet. So to use such a service like Telenet you need a modem to access the network, a PAD (packetizer) and HP's X.25 interface. Your terminal logs into the packet switching network who connects you to your computer. Packets of data from your terminal are sent to the HP who de-packetizes it and processes the data. The major advantage of X.25 is that it is the cheapest alternative for medium to high volume sites.

In 1982, our company was chosen to be a BETA test site for X.25. We were looking into these options for our long term telecommunication needs for our nationwide network. The BETA test went relatively well. The speed left a little bit to be desired, but then the access node could only be 1200 BAUD or less. This made the effective communication rate 1200 BAUD. HP did not support block mode in X.25. This means any full screen editors or V/3000 screens could not be used. Guess what? All of our systems were developed using V/3000.

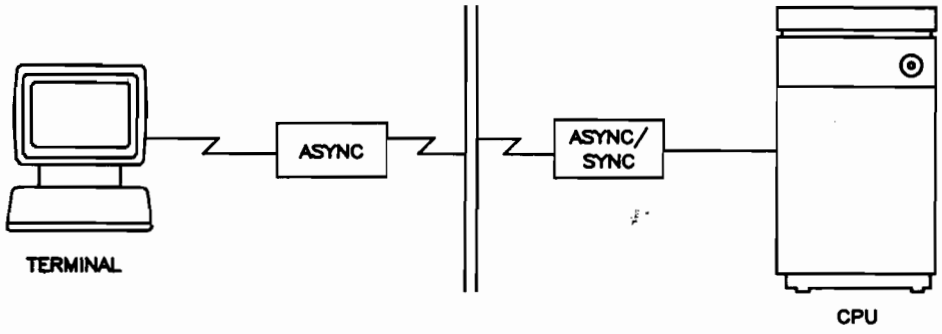
We decided not to use X.25 presently based on a number of items:

1) X.25 did not support block mode screens. Now you can use HP's PAD, the HP/2334 controller for block mode use
2) X.25 access was only at that time 1200 BAUD (Telenet said then that 2400 and 4800 BAUD would be available late 1983 and 9600 BAUD would be available in 1984.)
3) Telenet closed their St. Louis office, which being we are a centrally supported site, made us uneasy in the area of support.

Despite all the concerns sited, I still think that X.25 will be the alternative of choice in the future.

Async/Sync Convertors

An async/sync convertor is a hardware device which allows an asynchronous terminal to communicate to a computer via a synchronous phone line hookup. These devices are readily available from most of the modem vendors. The devices that we evaluated came from ATT and MICOM. The way these devices work is you attach the device between your terminal and the modem, and on the computer the device is attached between the computer and the modem. The device takes asynchronous protocol and converts it to synchronous protocol for transfer down the synchronous phone line. When the data arrives at its destination the data goes through the modem into the async/sync convertor where it is converted into asynchronous protocol. (Figure 1)



According to the vendors, this whole process should be transparent to the user. This means that you should notice no change in response time. When the vendors demonstrated these convertors to us, they hooked it up to my terminal which is hardwired to the computer and operates at 9600 BAUD. The vendors installed a "pseudo synchronous phone line" to the computer at 9600 BAUD and then hooked up my terminal to the async/sync convertor. Despite the claims of the vendors, I did notice an appreciable difference in the response time. This was a serious drawback to us because of the printing involved via a slaved printer, which I discussed earlier.

MTS/Synchronous Phone lines

MTS is the software that HP markets which allows HP terminals to communicate via synchronous phone lines. To use MTS you must have the following in place:

1) MTS software
2) An INP board for every physical line coming in (remembering that you can multidrop phone lines or have multiple users per phone line)
3) Install a POD or a 2333 controller at each site or terminal that you wish to be able to communicate via the MTS network.
4) Reconfigure your terminal

The INPs, PODs or HP/2333 controllers and MTS software is a one-time cost. A POD only allows one device per node to communicate to the computer. A HP/2333 controller, which is HP's version of a multiplexor, allows you to hook up multiple devices per node on your synchronous phone line. For an example, you could have a printer (non-spooled) and a terminal or a terminal, a printer, and an HP/150. The controller is expandable up to 16 devices.

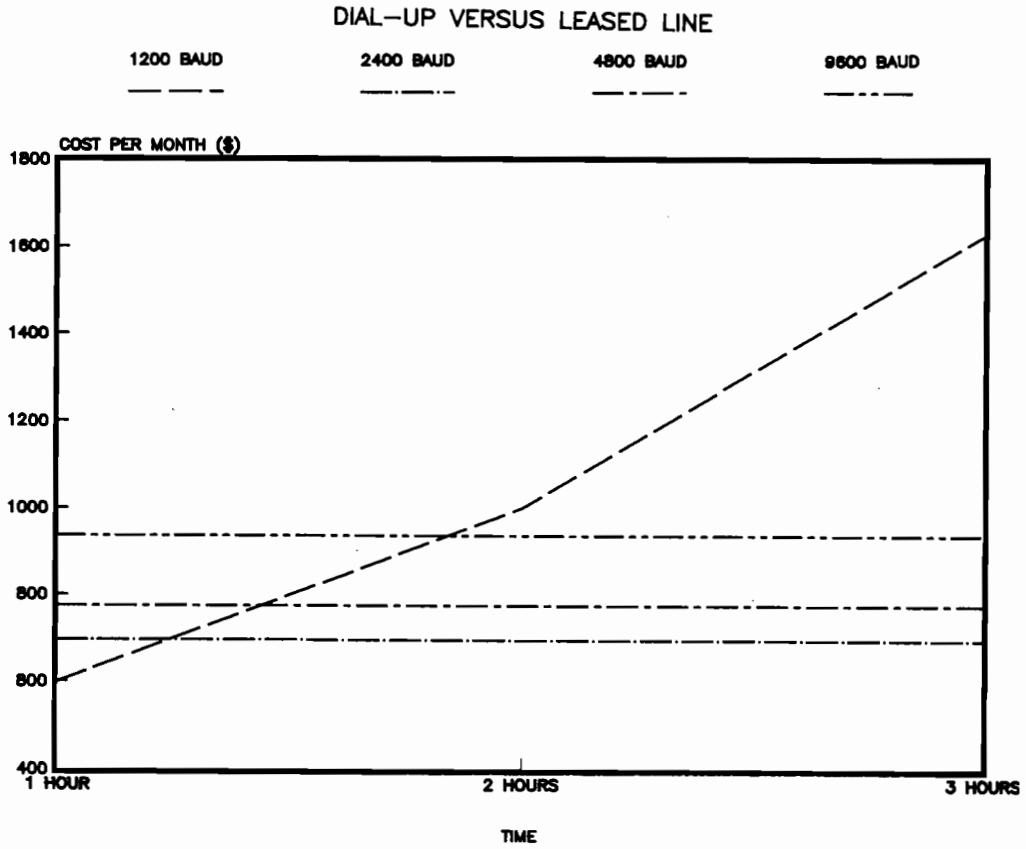
The MTS software is a proven piece of software which does not have any major bugs that I am aware of. It is HP which means that we still are only dealing with two vendors, ATT and HP. The operation of the MTS software is transparent to the user. When the POD was hooked up to my terminal, I could not tell that it was in place. The machine acted the same, except for displaying my password which can be fixed via a change to the command file. When and if we decided to expand our network, all that would be necessary would be to configure the devices into the MTS configuration and hook up the POD. With MTS, you can also assign a polling sequence, thus allowing you to give more time to your active users or devices.

Our company elected to use this option with the HP/2333 controller so that we were able to unslave our printers and run them hot. This increased their print speed up to the capacity of the printer. It also allowed the user to operate two devices at one time, i.e. enter data in on their terminal and print a report. If we had elected to use any of the other options, we would have had to buy a separate device for each terminal we

would have wanted to operate. This would have gotten very expensive, very quickly.

Cost Comparisions

Most people, in addition to the relative functionality of each option, are also interested in the cost comparision of each option. The figures that I am providing are the costs at the time of comparision, namely late 1982. This is fine for my objective, which is to show the relative cost of each option, rather than the cost to install a network today. For all of the the non-dialup options the relative phone line costs are the same. So for your review, this chart shows you the cost of operating a dialup line at 1200 BAUD at a distance of 300 miles and the cost of operating dedicated lines or leased sync lines at 2400, 4800 and 9600 BAUD at the same distance.

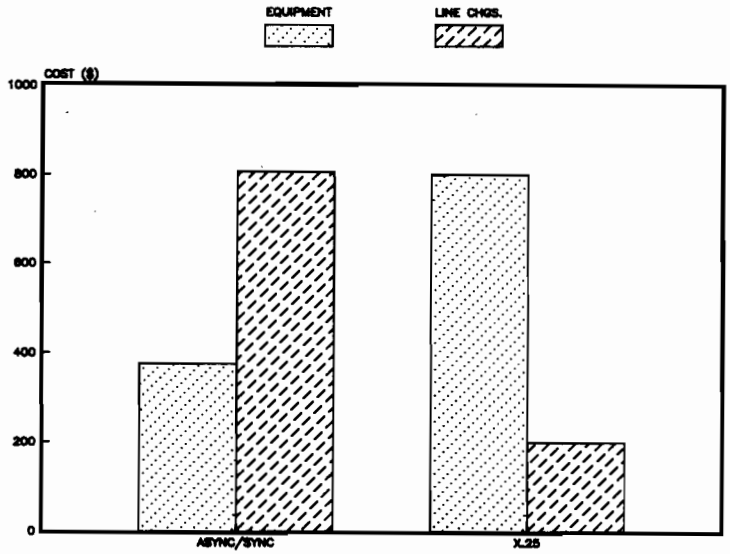


You can see from this diagram that the breakeven point for dial-in lines is about two hours of use per day. You can also see for 24 hour access on a leased line that you pay no more that five minute access. You can also see that there is a premimum for speed. The price of all data lines is calculated by adding up the cost for:

1) Data equipment - modems, async/sync convertors
2) Mileage
3) Type of service - leased line at a certain BAUD rate or dialup.

The comparison that follows for X.25 versus ASYNC/SYNC convertors and MTS assumes that the phones lines are in place at whatever BAUD rate that you elect. (Figure 3)

ASYNC/SYNC VERSUS X.25



You can see from this diagram that the least expensive alternative for volumes over two hours a day is X.25. This is not surprising because of the way it works. It is more efficient.

Each of you will have to evaluate which of the telecommunications options best suits your needs. I only hope that I have given you a look at what is available.

V. Futures

In the next year, we are going to embark on a new phase of our network project. Currently, we are not in a pilot stage, but in a full production mode, with two part-time operators and a full-time system manager. The project now has four full time analysts working on the project and two SUPER-USERS providing day-to-day application support. Our overall staff has grown from eight to thirty-two professionals. Our hours of operation for the network are from 12:00 A.M. to 8:30 P.M. and then from 9:30 P.M. to 12:00 A.M.

Next year we plan to install another twenty-eight sites bringing us to thirty-two sites across the United States. In addition, we are thinking of installing sites in Canada. With the addition of these sites, we were faced with making some hardware decisions about the kind of terminals to install at the new sites. We could stay with the tried and true HP/2624, but this provides limited functionality. We could also install micros that can communicate to the HP/3000 for use of the applications library, plus allow the use of stand-alone applications like CONDOR (database), VISICALC, and WORDSTAR (word processing). The ability to use this terminal for the satellite office's unique uses well exceeds the extra cost of the micro computer. So, we have ordered 28 HP/150 to install next year in more satellite offices. I am sure that the challenges that face us in the area of training and support greatly surpass even my active imagination. Despite all the challenges that await the MIS staff in this endeavor, I see areas of opportunity arising, such as electronic mail. These are exciting and rewarding to both the MIS staff and our users.

VI. Conclusions

I guess I had two objectives that I wanted to accomplish in this paper. The first was to acquaint you with some of the telecommunications alternatives that are available to you. The second was I wanted to give you some hints on what not to do when setting up a network. I have learned a great deal in two years of working on this network project, and I would like to share some of the things that I have learned.

DO NOT EVER DO THIS

1. Install a system without proper environment controls, especially clean power. Telecommunications interpretes changes in fluxuation in power as data. Not a pretty sight. Bad power (i.e., not clean and grounded) can cause failure in the telecommunication equipment and the computer itself. Power problems are those types of elusive problems that you can live without.

2. Install a system without a person to take care of it. The user will only complain about the lack of such a person, when you lose two weeks of data due to a system failure and because no one has had time to do backup.

3. Install a pilot for a network without benchmarks and a definite end to the pilot phase. What happens when you do this is that your users have no way of evaluating success or failure and then want to extend the pilot period. This makes MIS planning for improvements and development difficult, if not impossible.

4. Install a network without easy access to a telecommunications person. It is a very, very, very hard job to pick up telecommunication's expertise on the fly. Telecommunications is weird magic anyhow. Get yourself a telecommunications magician; it will make solving those telecommunications problems a lot easier. If you think, what problems, you are kidding yourself. We have telecommunications problems at least three to four times a week.

5. Install a network without an expansion plan. The network will not remain static, despite what your user says. At the very least the technology will change, and you will want to take advantage of those changes.

6. Buy your telecommunications equipment. The technology is changing so fast in this area that you should not buy, but lease.

7. Lose your copy of Ross Scruggs TERMINAL TALK. It will save your soul on that wet and nasty night that you are there reconfiguring the system for the new telecommunications lines. Following his suggestion will keep smiles on your face and your user's face.

8. Take this paper as total fact. This paper is only my experience and is not meant to be the definitive paper on network design. I am not a telecommunications whiz. I am only a lowly analyst that happened through fate to get involved in installing a nationwide network.

TRY DOING THIS

1. Online documentation. We have found that most users will not use the hard copy documentation, but they will use online documentation. A style which we have found is the best is like the help facility in EZCHART.

2. Train the trainer. It is impossible for one or two people to train 150 users in the time period needed, so we have used our SUPER-USERS for such training.

3. Have the user assist in writing user documentation. We have found that the user writes the best user documentation, but we have also found that they won't write user documentation. So we are still writing user documentation, but getting the user to assist us in the editing of the document.

4. Accept obsolescence. No matter how good your plans are, technology and user needs change so rapidly that your best plans will be obsolete in 12 months.

5. Solict upper management support. Grass roots campaigns take a long time to get their message across. Try upper management instead. When upper management speaks, it is like E. F. Hutton, everyone listens.

I hope this paper was of some use to you.

P.S. I want to thank all those who helped me in putting this paper together. Thanks to:
Ray Geiger, Tom Puorro, Dan McFall, Jan Markovich,
Ron Helms, Dave Bennett, Michael Aguera

ENHANCING FORMS

by Alvin Bruce Charity, I
and
Katherine Joan Dante

ABSTRACT

Screens can make or break a system. Attractive screens help sell the system to the user. Neat, logical screens can increase data entry rates, improve data entry quality, and present that "professional" image to the user. This paper will demonstrate how to turn screen enhancements on and off programmatically, and how to create quality screens using the line-drawing character set.

Examples will be provided which demonstrate the improvement in screen legibility and show how to incorporate the necessary escape sequences in COBOL, FORTRAN, and SPL programs.

This paper is meant for programmers and analysts who use VPLUS and want to improve the legibility of their screen design.

PART I USING FORMSPEC

INTRODUCTION

We hear the term "user-friendly" a lot: user-friendly systems, user-friendly computers, user-friendly screens. When we try to use these "user-friendly" what-evers, we quickly find that "user-friendly" is as hard to understand as non-user-friendly. A truly user-friendly screen would offer the user a day off with pay!

We aren't going to show you how to make user-friendly screens. That's like making good-tasting medicine. If it is done right, you then have to put on child-proof caps. No, what we are interested in is legible screens. Screens that don't scramble your eyeballs when you try to read them. If the screens are used to capture data from pre-printed forms, we want them to resemble those forms as closely as possible (and have you ever seen a user-friendly form?)

LINE-DRAWING

One thing that amazes me about Hewlett-Packard is its lack of bragging. If you do not happen to read the manuals that come with the terminals, you might not know that a line-drawing character set even exists. And the only way to find out whether it works with FORMSPEC or not is to use it.

We tested it out, found that it worked, and began to use it to improve our forms.

The first system we installed on our HP was a personnel system. Now, as everyone in the federal government knows, our personnel system is fueled with the Standard Form 50, familiarly called the SF50. Nothing can be done without first filling in a request for a personnel action, so that the personnel

office can respond with the SF50. The National Science Foundation has been producing these SF50's from the computer since the seventies.

Our first data input screens, designed on a Honeywell 6060N, were extremely primitive, formatted more for the convenience of the computer than for human beings. Data elements were listed on the left of the screen and the correct values were entered on the right. Errors were indicated with an asterisk or error code by the data element name.

On moving to the HP, we tried to make the screens look as much like the form as possible. However, without lines to separate the different parts of the form, the screen was confusing. We tried separating the elements using the line-drawing character set and--Voila! a clean screen, easy to read and equivalent to the actual form. Because the lines are distinct from the written material, they can be used without extra spacing. This helps minimize the number of screens needed.

ACCESSING THE LINE-DRAWING CHARACTER SET

Alternate character sets can be accessed either through function keys or through escape sequences. On some terminals, they can be accessed only through escape sequences. The sequence <ESC>)B will make character set B, which is usually configured as the line-drawing set, the alternate character set. Your terminal's manual will show you the proper keys for various line formats. Although many terminals treat upper- and lower-case alphabetic characters the same when translating to the line characters, use only upper-case alphabetics for those terminals which are case sensitive. Once you have established the alternate character set, use <CTR>N and <CTR>O to switch between it and the ASCII set. (N for new, O for old.) You can flip between <CTR>N and <CTR>O as often as necessary to get the exact effect you want. Remember that a line at a time is converted, not the entire screen.

If your terminals do not have the line-drawing character set, ask your HP representative for the simple field upgrade that will provide it.

ENHANCEMENTS

Perhaps because Hewlett-Packard started out making tools for scientists, it appears to leave much of its capability for the user to discover. Fortunately, our division is blessed with several people who, even under tight deadlines, have the curiosity to try various HP features in new situations. For ex- ample, FORMSPEC mentions underlining, half-bright, blinking, and inverse video, but only for the data entry fields. It was up to us to find that these could also apply to screen text.

To make matters more difficult, the HP2645A, sold as the program- mer's terminal, does not give you access to these screen enhance- ments through the function keys; while the HP2624B, billed as the data entry terminal, does.

Interestingly, when using FORMSPEC on the HP2624B, you can gain access to these enhancements only while designing a screen! It seems that HP did intend these to be used in form design, it just was not made clear in the HP documentation.

When using a terminal without function key access to the screen enhancements, use <ESC>&d followed by the appropriate code chosen from the table given in the "Display Enhancements" section of your terminal manual. A more detailed look at these escape sequences will be given later.

The first use of these enhancements was to emphasize parts of a form. Even with the line-drawing set, the legibility of forms can be improved with full- and half-bright inverse video. We first used full-bright inverse video only to high-light sections of a form. (Blinking, unlike other enhancements, appears to ir- ritate the user more than help.)

SCREEN STANDARDS

With our expanded knowledge of HP capabilities, we began to standardize our screen formats. Our current standards require the screen be divided into three sections:

1. A title line, which defines the screen and provides date and time

2. A data entry or retrieval portion, which displays data and asks for input

3. An instruction portion, which tells the user which key to press and what data to enter

The different sections of the form are separated with lines. The instructions are emphasized with the use of half-bright inverse video, set up in the three columns: TO do, ENTER, and PRESS, with each column heading underlined. (In some cases, space can be saved on the form, without losing legibility, by dividing the in- struction section in the middle of the screen, putting directions in both halves.)

The title line will be helpful if there are ever any problems, undiscovered bugs, in the production programs. It contains the date and time the screen was invoked, as well as the title of the system and a screen identification which enables the programmer to find exactly where the program may have gone wrong--of course, our programs never fail!

The instruction section solves the problem of not all Hewlett-Packard terminals supporting the display of function key labels. (In fact, the ones we supplied our data entry people, the HP2645A's, lack this feature.)

CONCLUSION TO PART I

Using screen enhancements and the line-drawing character set in FORMSPEC, we have seen how to produce clean, legible screens. In the next part of this paper, we will learn how to change the en- hancements of the data-input fields in application programs.

PART II PROGRAMMATIC CONTROL OF TERMINAL ENHANCEMENTS

As just described, the terminal enhancements are alive and well. Now that we know they exist, let's take it a little further. Did you know that these features can be programmatically controlled? Well, the answer is they can! So now we want to make you the programming "Gurus" that you think you are. How can this be done? Well, it's a piece of cake.

Most often online applications are designed using character prompts or a screen writer package. For this discussion the screen writer package is VPLUS (V3000). There are occasions in many online applications where it's desirable to turn on and off data field enhancements. This feature gives the online applica- tion what we consider to be "Terminal Special Effects". These terminal special effects can show various ways the information is displayed to the user:

1. required data values,
2. required data values conditionally based on other data values,
3. data values in error,
4. key values for the process or
5. other reasons as defined by the application.

Using terminal enhancements, we can add a touch of "pizzazz" to the user/computer interface. But remember, nothing is free in this lovely world of computers; it will cost just a little in computer resources to employ these fancy features. However there should not be any noticeable change in the user response time.

WHAT ARE THE TERMINAL ENHANCEMENTS?

Terminal Enhancements are a standard feature of your HP ter- minals. These enhancements are ways to display characters and background, which consist of the following:

- Security Video - the character display is suppressed when entering or displaying data to the terminal. This enhancement is used in

conjunction with fields for which passwords or similar security-sensitive data must be entered through the keyboard.

- Inverse Video - black characters are displayed against a full-bright white background.
- Half-Bright Video - characters (or background for inverse video) are displayed at half intensity.
- Blinking Video - characters repetitively blink on and off.
- Underline Video - characters are underscored.

These enhancements may be used separately or in any combination. When used, they cause control bits to be set within the display memory of the terminal. If the content of the display memory is subsequently transmitted to a host computer, these control bits are translated into escape sequences which are transmitted along with the displayable text characters.

From the keyboard, you can enable and disable the various video enhancements using the "enhance video" set of function keys. This is accomplished by pressing the "AIDS" function key followed by the enhance video function key.

Programmatically these enhancements can be enabled or disabled by embedding escape sequences within the data. The general form of the escape sequence is as follows:

<ESC>&d<enhancement code>

where the enhancement code is an @, s, or S or one of the upper- case letters A through O specifying the desired enhancement(s). Some of these enhancements are defined as follows:

- Inverse Video = ESC&dB
- Half-Bright Video = ESC&dH
- Underline = ESC&dD
- Blinking = ESC&dA
- Security = ESC&dS
- Security with the Inverse Video = ESC&dsB
- End enhancement = ESC&d@

Additional information about the terminal enhancement features can be obtained in the "DISPLAY CONTROL" section of your HP ter- minal manual.

HOW TO PROGRAMMATICALLY INVOKE TERMINAL ENHANCEMENTS

Various methods for turning on and off the video enhancements will be discussed. For the sake of simplicity most of the em- phasis is focused on the inverse and half-bright video. Let's start by setting the stage with a sample application.

Sample Application

This sample application is for a personnel system. Our task is to capture and update employee data using an employee-data form. The data contained on the form will be:

1. Last-name
2. Rest-of-name
3. SSN
4. Sensitivity
5. Date-of-birth
6. Veteran-preference
7. Sex
8. Citizenship
9. Effective-date
10. Position-code
11. Nature-of-action-code

All of this data is required at initial capture time but can be updated later.

CASE 1 - USING ENHANCEMENTS WITH VPLUS AND NO ESCAPE SEQUENCES

The Case 1 example will use a VPLUS (V3000) application to ini- tially capture the employee data. All required fields will be shown in inverse video. Once the required data is captured from the form, the data will undergo edits. Data values that don't pass the edit criteria will remain in inverse video, while values that passed will be set to half-bright inverse video on the screen.

One way to show the required data fields in inverse video is to define the employee-data form in VPLUS via FORMSPEC. The draw- back of this method is that the fields will remain in inverse video after the data has passed edits, which counters the philosophy behind the use of inverse video versus half-bright in- verse video. Since we want the required data values to be placed in half-bright inverse video after passing the edits, we can ac- complish this by defining all data fields as half-bright in VPLUS and programmatically controlling the inverse video when needed. The VPLUS error enhancement will be set to inverse video via FORMSPEC.

Case 1 suggested programmatic code:

The following code consists of only the VPLUS intrinsics to show how to paint (or print) the employee data form to the user's ter- minal. This scenario assumes that the form name and other data have already been initialized.

1. CALL "VOPENFORMF"
2. CALL "VOPENTERMINAL"
3. CALL "VGETNEXTFORM"
4. CALL "VINITFORM"
5. CALL "VSHOWFORM"

At this point the terminal is painted with the enhancements as set in VPLUS via FORMSPEC--half-bright inverse video. This is not what our specifications called for.

The following shows two additional VPLUS calls that can be used to turn on inverse video for all required data fields.

1. CALL "VOPENFORMF"
2. CALL "VOPENTERMINAL"
3. CALL "VGETNEXTFORM"
4. CALL "VINITFORM"

4.1 CALL "VEDITFIELDS"

4.2 CALL "VPUTWINDOW" USING COMAREA, MESSAGE, MSGLEN. (where MESSAGE contains all blanks)

5. CALL "VSHOWFORM"

The execution of the VEDITFIELDS intrinsic will cause all required data fields to be flagged with an error, since the fields have been defined in the form as required data and no data has been entered. The VEDITFIELDS intrinsic uses the error en- hancement as defined in VPLUS which is inverse video. The use of the VPUTWINDOW intrinsic will suppress or blank out any error message to the user at this time.

If any data errors occur during the user's entry process, only those data fields in error will remain with the inverse video en- hancement. The data fields that passed the edits will be placed in half-bright inverse video.

This example illustrated the use of the inverse and half-bright video with VPLUS and requires no additional programming using the escape sequences.

CASE 2 - USING ENHANCEMENTS WITH VPLUS AND THE ESCAPE SEQUENCES

The Case 2 example will require the program to conditionally turn on and off the enhancements based on certain data statuses. For example if the nature-of-action code changes on the employee-data form, additional data must be updated in our personnel system. This will require a second form, entitled the nature-of-action form, be displayed. This form will consist of:

1. Name-of-office
2. Location-of-office
3. Position-title
4. Salary
5. Grade

When the nature-of-action code is a "P" (for promotions):

- position title

- salary

- grade

will require updates. This means that these data values should be highlighted with the inverse video enhancement prior to the form being painted.

On the other hand, if the nature-of-action code is a "D" (for detail assignments), this will require:

- name-of-office

- location-of-office

to be updated, and they should be displayed with the inverse video enhancement prior to painting the form.

For both of these situations, the program will retrieve the data from the personnel data base for the nature-of-action form prior to painting the screen. Again, once data values pass the edits, any subsequent form-painting should be in half-bright video for those fields only.

Case 2 - Suggested VPLUS Interface

One solution is to have separate forms showing the different fields with the inverse video enhancement via the FORMSPEC definitions; but we need to set them to half-bright for sub- sequent painting once they pass the edit criteria. Also, using multiple forms to show the different enhancements could possibly cause our forms file to get extremely large, especially for large or complex applications.

The following code depicts the VPLUS calls required for painting the nature-of-action form to the terminal:

1. CALL "VGETNEXTFORM" (using the data retrieved from the personnel data base)

2. CALL "VPUTBUFFER"

3. CALL "SHOWFORM"

By adding the VSETERROR intrinsic, we can programmatically turn on the inverse video enhancement for those data values requiring update. For example, the nature-of-action-code "D" change would require the following fields be enhanced:

- name-of-office - field #1

- location-of-office - field #2

The VPLUS calls would then look like this:

1. CALL "VGETNEXTFORM"

2. CALL "VPUTBUFFER"

2.1 CALL "VSETERROR" USING COMAREA, FIELD#1, MESSAGE, MSGLEN

2.2 CALL "VSETERROR" USING COMAREA, FIELD#2, MESSAGE, MSGLEN (where MESSAGE contains all blanks)

3. CALL "VSHOWFORM"

Resulting from this, the name-of-office and location-of-office will be shown with the inverse video enhancements and the cursor will be positioned at the first field requiring a data change.

Case 2 suggested code using VPLUS and escape sequences:

Using the escape sequences to turn on inverse video for the Case 2 example would require the data buffer used for the VPUTBUFFER to carry these enhancements. As VPLUS users already know, the VPUTBUFFER is a mirror image of the VPLUS form. So now we can define a field or fields with the enhancements we want to use. This can be accomplished as follows:

- Inverse-video-enhance value "ESC&dB"

- Half-bright-enhance value "ESC&dH"

In our program we would:

```
COBOL   - move inverse-video-enhance to field#1, field#2
```

```
SPL      - field#1(*)inverse-video-enhance
           field#2(*)inverse-video-enhance

FORTRAN  - field#1=inverse-video-enhance
           field#2=inverse-video-enhance
```

In this solution, the cursor will be positioned at the first changeable data field and not necessarily at the first required data field. (In this example, the first changeable data field is the first required data field.)

The examples presented are just some ways to use the enhancements for terminal special effects. Using your own creative ingenuity and further research, you will find many ways to enhance applica- tions using these features.

Pretty user friendly huh!

Don't worry, as you use these concepts, they will become programmer-friendly.

HP 3000 RPG/VIEW
INTERACTIVE PROCESSING TOPICS

Christopher J. Colburn

This paper is geared towards the beginning RPG programmer on the HP3000 and is intended to demonstrate how interactive data handling programs can be created on the HP3000 by using the RPG/View interface. This short paper includes a sample screen program in Appendix A which is presently in use at the G.D. Armstrong Company. The author wishes to state that the program may or may not be the best written RPG program available and any comments or suggestions by the readers of this article would be greatly appreciated.

The author also assumes that the readers of this article have a good knowledge of both RPG and VIEW/3000.

INTRODUCTION

Have you ever purchased a product from a department store, gotten it home, and become frustrated because the product was harder to use than it was first anticipated? If so, and if you are an RPG programmer working on the HP3000 series of computers, then this paper is dedicated especially to you!

Even though documentation in the RPG Manual on the RPG/View interface is very through when dealing with batch programming, the discussion of how to apply interface to interactive programming is very lax. The example provided for programmers in the manual is the RPG version of VIEW/3000's "ENTRY" program, but this does not show an entry-level RPG programmer how to write a program which will search files, update existing records, display existing fields of a record, and add records to a file. Since documentation on creating interactive screens on the HP3000 in RPG is sparse, and because I feel it is unfair to put anyone else through the misery of searching through manuals and testing many programs, I have decided to write this paper on using VIEW/3000 and RPG on the HP3000 series of computers. I have also included an example in Appendix A to help the first time RPG programmer utilize the interface for many programming needs.

When designing a good computer system, the programmer/analyst must try to make it user friendly. Since an operator/user will have most of the day-to-day interaction with the hardware and software on the system, it is imperative that the entry of data be designed in such a way as to encourage productivity. One way to do this is to make the user completely comfortable entering data into the system.

There are several ways to enter data on the HP3000. Data can be entered through the utilities of the programmming langugage used, through the V/3000 ENTRY program, or interactively using a combination of programming language utilities and V/3000.

Programming languages like COBOL and RPG have commands which allow the user to enter data into the system by the programming language. For example, COBOL uses the "ACCEPT" commant to accept data from the user at the terminal. RPG uses the "DSPLY" command to enter data through the system. These commands are good to use when entering in one or two fields of information, or if a small number of people are using a particular system, but are limited and slow when handling large amounts of information about a record or when a large number of people are using the system.

The "ENTRY" program provided by Hewlett Packard for use with V/3000 is

adequate and easy to use, but has major faults. On one hand, it provides for easy maintenance of screens without the writing of bulky programs. Editing of screen fields is also handled by VIEW. All items are keyed into a batch file, reformatted, and then placed into a KSAM, IMAGE, or MPE file. ENTRY, however, has overriding negative factors to discourage its use. Some of these factors are that ENTRY does not handle data interactively (on-line processing), ENTRY only updates records within the batch file, and ENTRY does not determine whether the record you are working on is in the main file until after reformatting.

The best kind of data entry system is one which will provide maximum data with minimum effort on the part of the user. This type of information system can be achieved on the HP3000 by using a combination of one of the various programming languages (FORTRAN, COBOL, BASIC, RPG, etc) amd V/3000. FORTRAN, COBOL, and BASIC allow the programmer to code in calls to V/3000 intrinsics which build the VIEW screens. RPG, because of its structure on the HP3000, has to internally call these intrinsics. Thus, it is not as easy to call V/3000 intrinsics for the RPG programmer as it is for programmers in other languages. Status words are not available the the RPG programmer, and calling V/3000 screens in RPG seems almost impossible.

Each language, however, has its tools to create screens on the HP3000, and RPG is no exception. Because chapter 13 of the RPG MANUAL (RPG INTERFACE TO V/3000) is documented in detail, I will only summarize the major functions of the RPG-V/3000 interface in this paper. These functions are illustrated in the sample program displayed in Appendix A.

GETTING STARTED

I started writing RPG interactive screen programs by playing with the batch RPG screen program, ENTRYRPG. I compilied that program, created a screen to enter test data, and keyed in some data. The program worked okay, but I wanted (and my company needed) something better. My company and I needed a dynamic data handling screen program which would call up a record in a file, check to see if the chosen record was on file to update it, or enter the new record. In short, we needed a program which would make full use of the capabilities of the machine we were using. Because the RPG/VIEW interface seemed only able to handle batch records, our company even looked at going with a second language such as COBOL or BASIC just to write our screen programs. The monthly maintenance was too expensive to run just these screen programs, and a total conversion from RPG to COBOL, or RPG to BASIC also was too time-consuming and costly for us. After examining the ENTRYRPG program, I realized that it was geared totally to the "batch" method of entering data. Since my company needed an interactive data handling screen program, and I had I had no examples from which to create one, I "robbed" the best parts of ENTRYRPG and started to create an interactive data handling screen using RPG.

As in COBOL handling of screens, "top down" programming in the RPG handling of screens is next to impossible. I did, however, try to "standardize" as much of the actions and events that I possibly could. For example, any action which is called from the program has been placed into a standard subroutine. Then, all I had to do is call that subroutine (through the "EXSR" command), and the action was executed.

I also found out that the editing of data (negative numbers, right justify, blanking fields, etc) was handled very poorly by the RPG/VIEW interface. If a negative number was entered into a numeric field, the program would accept the negative number. However, if you were going to look at this field when updating the record, the interface will overlay the negative sign over the last digit on the right. V/3000 will not accept the number as numeric. Because of this I decided to define numeric fields which could be both positive or negative as character fields. I added 2 characters to each field (one for a sign, one for a decimal point), and created my own edit subroutines on all fields going to or coming from a screen.

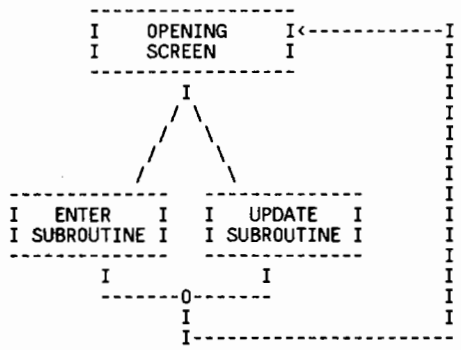
CREATING THE INTERACTIVE SCREEN

First, as a point of review:

The RPG/VIEW interface does not strictly follow the RPG logic cycle. All indicators must be set on or off manually before any action is performed. The EXCPT, READ, and CHAIN commands dominate this type of program. Subroutines to edit fields going to or coming from the screen are needed, for the RPG/VIEW interface handles negative numbers very poorly. This interface also uses what are called "ACTIONS" and "EVENTS". ACTIONS are output records written to the WORKSTN file. EVENTS are input records read from the WORKSTN file. More information about ACTIONS, EVENTS, the RPG/VIEW interface is available in Chapter 13 of the HP3000 RPG Manual.

One of the first things to do in the design of a screen is to try to picture what the screen is going to do. In the example provided (Appendix A), a screen was needed to look at an Accounts Receivable entry file. The file was a KSAM file, so I knew that I would need a first

or "front" screen with the key field on it. Once the key was entered on this screen, I knew that the record had to be added to the file (ENTER subroutine) or had to be updated (UPDATE subroutine). After the record was added or updated, the program was to return to the "front" screen and await either entry of another key or termination of the program. I pictured it like this:



Now that I had an idea of what course my program was to follow, I wrote it, following these basic steps:

1. Define the input/update files in the "F" specs.

2. Define the single WORKSTN file (TRANSFILE) on the "F" specs.

3. Set up edit field and message arrays in the "E" specs.

4. Define the input/update file fields on the "I" specs.

5. Define the screen fields on the "I" specs. Multiple screens are like a file that has different types of records; each type of record is defined on the input specs. Similarly, each different screen is defined on the input specs.

6. Define WORKSTN function keys.

7. Bring first screen up and decide whether the record is to be entered or updated (refer to Appendix A).

a. Move form name to form name field.

b. Set up the correct repeat and freeze form options.

c. Move form name field to FORMSPEC NEXTFORM option (EXSR INTFRM).

d. Specify next form name and repeat/freeze options (EXSR CHGSUB).

e. Set the next form from forms file, set repeat/freeze options, and initialize fields according to FORMSPEC (EXSR INTSUB).

f. Reset the form - clear the fields (EXSR GETSUB).

g. Replace data in V/3000 data buffer with data from user program buffer (EXSR PUTSUB).

h. Display current form, initial data, and messages. Read input from terminal to V/3000 data buffer and read WORKSTN record (EXSR SHWSUB).

i. After the screen is read, clear the message window (EXSR CLRMSG).

j. Check to see if either a function key or the ENTER key was hit. If an invalid function key was hit, a message will notify the operator. If ENTER was hit, perform edits specified for the fields in the current form. This is followed by a read of the WORKSTN file to check for errors (EXSR EDTSUB). If errors are found, go back to "h" above and try again.

k. If the screen passed the edits, then write the data from the data data buffer to the user program buffer, then do a read of the WORKSTN file (EXSR GETSUB).

l. Move any important screen data which you do not want to lose to a hold field or area.

m. Move user program buffer to the data buffer (EXSR PUTSUB).

n. Perform a chain on KSAM files to see if you are going to enter or update the record. Then enter or update the record.

o. Set up the first screen again and set off all indicators. Then go back to the beginning of the cycle and start all over again. If you wish to end the program, function key "F8" has been coded to end the program.

8. When entering or updating a record, you follow a similar pattern to the one just described in step 7. When updating, however, you must first move the fields from the record being updated to the screen and then perform the actions previously described.

9. Also: In the sample program in Appendix A, note that the WORKSTN file (TRANSFIL) has a continuation ("K") record. Unless you define your VIEW form as "FORMA", you will need an MPE file equation to run this program. This equation would be similar to:

:FILE FORMA=GDAR4FM.PUB.GD

Finally, to summarize this paper, I wish to state that effective interactive screen programs on the HP3000 can be created with the RPG/VIEW programming interface. Because, in my opinion, the documentation in the RPG Manual did not properly address interactive on-line programming, I felt that this paper was necessary. At the very minimum, a working example has been provided with this paper, and it is the author's hope that RPG programmers can benefit from his labor on this topic. Hopefully, this information will lead towards standardized, well written, and easily maintainable RPG programs on the HP3000.

********** APPENDIX A **********

```
HDUMPFILE
F**-------------------------------------------------------------**
F** THIS SCREEN PROGRAM UTILIZES SUBROUTINES TO CALL            **
F** ACCOUNTS RECEIVABLE SCREENS. FORMA GDAR4FM ENTERS,           **
F** UPDATES, OR DELETES A/R GDENTER RECORDS.                     **
F**  CHRISTOPHER J. COLBURN, GD ARMSTRONG COMPANY  082583.       **
F**-------------------------------------------------------------**
F** DEFINE INPUT AND SCREEN FILE LENGTHS                        **
F**-------------------------------------------------------------**
FGDENTER UC  F 128 128R08AI     3 DISC                          A
FTRANSFILUD  V     299            WORKSTN   L  5
F                                               KFORMS FORMA

E*-----------------------------------------------------*
E* SET UP PROGRAM EDIT FIELD AND MESSAGE ARRAYS       *
E*-----------------------------------------------------*
E                    CARA        9  1
E                    ARA         7  1
E                    ARA1        7  1
E                    ARA2        6  1
E                    MSG1    1   1 60
E                    MSG2    1   1 60
E                    MSG3    1   1 60
E                    MSG4    1   1 60
E                    MSG5    1   1 60
E                    MSG6    1   1 60
E                    MSG7    1   1 60
E                    MSG8    1   1 60
E                    MSG9    1   1 60
E                    MSG10   1   1 60
E                    MSG11   1   1 60
E                    MSG12   1   1 60
E                    MSG13   1   1 60
E                    MSG14   1   1 60

I*-----------------------------------------------------*
I* DEFINE INPUT/UPDATE FILE FIELDS                     *
I*-----------------------------------------------------*
IGDENTER AA  01
I                                        1   2 GERCDE
I                                        3  10 GEREFN
I                                       11  160GEDATE
I                                       17  24 GEITEM
I                                       25  49 GEDESC
I                                       50  550GECUST
I                                       56  80 GENAME
I                                     P 81  823GETAXP
I                                       83  830GEC0D
I                                       84  840GETAXC
I                                       85  850GEPAID
I                                       86  860GECPST
I                                       87  870GEIPST
I                                       88  92 BLANK1
I                                       93  930GEINVC
I                                       94  950GEDL0C
I                                       96  970GE0C
I                                       98 121 BLANK2
I                                     P 98  1001GEQTY
I                                     P 101 1034GEPRIC
I                                     P 104 1064GEUCST
```

```
I                                       P 107 1094GEPRPG
I                                       P 110 1113GEMPCT
I                                       P 112 1133GEFPCT
I                                       P 114 1172GEEXTN
I                                       P 118 1212GERTAX
I                                       P 122 1242GESTAX
I                                       P 125 1282GETOTL
I*-------------------------------------------------------*
I* DEFINE SCREEN INPUT FIELDS                           *
I*-------------------------------------------------------*
I***** ENTER KEY (00)
ITRANSFILAA  12   1 C0   2 C0   7 C7
I       OR        1 C1   2 C0   7 C7
I                                        3  17 FORMB
I                                       18  21 DATALN
I                                       22  29 SADATA
I                                       22  29 SAREFN

I        XB  13   1 C0   2 C0   7 C8
I       OR        1 C1   2 C0   7 C8
I                                        3  17 FORMB
I                                       18  21 DATALN
I                                       22 187 SCDATA
I                                       22  23 SCRC0D
I                                       24  31 SCREFN
I                                       32  370SCDATE
I                                       38  45 SCITEM
I                                       46  70 SCDESC
I                                       71  760SCCUST
I                                       77 101 SCNAME
I                                      102 1043SCTAX
I                                      105 1050SCC0D
I                                      106 1060SCTAXC
I                                      107 1070SCPAID
I                                      108 1080SCCPST
I                                      109 1090SCIPST
I                                      110 114 SCBLNK
I                                      115 1150SCINVC
I                                      116 1170SCDL0C
I                                      118 1190SCGE0C
I                                      120 126 SCQTY
I                                      127 133 SCPRIC
I                                      134 140 SCC0ST
I                                      141 147 SCPRCG
I                                      148 1503SCMPCT
I                                      151 1533SCFPCT
I                                      154 162 SCEXTN
I                                      163 171 SCRTAX
I                                      172 178 SCSTAX
I                                      179 187 SCTOTL
I*-------------------------------------------------------*
I* DEFINE FUNCTION KEYS. IN THIS PROGRAM, F2 DELETES    *
I* THE RECORD AND F8 EXITS THE SUBROUTINE OR PROGRAM     *
I*-------------------------------------------------------*
I        BB  78   1 C0   2 C8
I        BC  77   1 C0   2 C1
I        BD  75   1 C0   2 C2
I        BE  76   1 C0   2 C3
I       OR        1 C0   2 C4
I       OR        1 C0   2 C5
I       OR        1 C0   2 C6
I       OR        1 C0   2 C7
I*** NUMBER OF EDIT ERRORS (09)
I        CC  79   1 C0   2 C9
I                                        3  17 FORMB
I                                       18  220NUMERR       74
```

```
C*-------------------------------------------------*
C* MOVE FORM NAME TO FORM NAME FIELD               *
C*-------------------------------------------------*
C           BEGIN     TAG
C                     MOVEL"GDAR7"   FORMB
C                     MOVE "0"       RPTAPP  1
C                     MOVE "0"       FRZAPP  1
C                     EXSR INTFRM
C*-------------------------------------------------*
C* SPECIFY NEXT FORM NAME AND REPEAT/APPEND OPTIONS *
C*-------------------------------------------------*
C                     EXSR CHGSUB
C           START     TAG
C*-------------------------------------------------*
C* GET THE NEXT FORM FOR USE AND INITIALIZE THE FIELDS*
C* NOTICE THAT 91 IS SET ON. THIS IS USED WHEN CALLING*
C* ACTION "PUTDTA", AND DETERMINES WHICH SCREEN IS  *
C* BEING PROCESSED.                                 *
C*-------------------------------------------------*
C                     SETON                     91
C                     EXSR INTSUB
C*-------------------------------------------------*
C* DO GETDTA HERE TO CLEAR/RESET SCREEN            *
C*-------------------------------------------------*
C                     EXSR GETSUB
C*-------------------------------------------------*
C* MOVE LAST KEY PROCESSED TO FRONT SCREEN TO TRACK *
C* WHERE OPERATOR IS.                              *
C*-------------------------------------------------*
C                     MOVE BFREFN    SAREFN
C*-------------------------------------------------*
C* REPLACE DATA IN V/3000 DATA BUFFER WITH DATA IN  *
C* USER PROGRAM BUFFER. (MOVE DATA FROM BUFFER TO SCN *
C*-------------------------------------------------*
C                     EXSR PUTSUB
C*  3 * DISPLAY FORM
C           REPEAT    TAG
C                     SETOF                     111213
C                     SETOF                     141516
C                     SETOF                     171920
C                     SETOF                     747522
C                     SETOF                     767778
C                     SETOF                     7966
C*-------------------------------------------------*
C*SHOW FORM AND PLACE INPUT IN DATA BUFFER.        *
C*-------------------------------------------------*
C                     EXSR SHWSUB
C*-------------------------------------------------*
C* 6  * CLEAR MESSAGE WINDOW                       *
C*-------------------------------------------------*
C                     EXSR CLRMSG

C*-------------------------------------------------*
C* SEE IF FUNCTION KEY OR ENTER KEY IS HIT HERE. IF F2*
C* OR F8 NOT HIT, THE KEY IS INVALID AND A MSG IS SENT*
C* TO THE OPERATOR. IF ENTER, CONTINUE.            *
C*-------------------------------------------------*
C   78                SETON                     LR
C   LR                GOTO EXIT                      EXIT IF F8
C   75                SETON                     76
C   77                SETON                     76
C* INVALID KEY (F1-F7) HIT - TRY AGAIN
C   76                MOVE MSG1      MSG    60
C   76                EXSR MSGD02
C   76                GOTO REPEAT
C*-------------------------------------------------*
C*  IF ENTER HIT - RUN VIEW EDITS                  *
```

```
C*-------------------------------------------------*
C                    EXSR EDTSUB
C*** IF ERRORS - RETURN TO REPEAT AND TRY AGAIN
C   74               GOTO REPEAT
C*-------------------------------------------------*
C* TRANS DATA FROM DATA BUF TO UP BUFF, THEN READ REC*
C*-------------------------------------------------*
C                    EXSR GETSUB
C* PERFORM USER EDITS
C          MORERR    TAG
C*-------------------------------------------------*
C* MOVE 1ST SCN INFO INTO HOLD AREA FOR DSPLY THE NEXT*
C* TIME THE FORM IS SHOWN.                          *
C*-------------------------------------------------*
C                    MOVE SAREFN    BFREFN  8
C                    EXSR PUTSUB
C          PCHAIN    TAG
C*-------------------------------------------------*
C* CHAIN KEY ON SCREEN TO SEE IF YOU ARE ENTERING OR  *
C* UPDATING THE RECORD. 22 ON IF REC NOT ON FILE.     *
C*-------------------------------------------------*
C          BFREFN    CHAINGDENTER              22
C                    SETOF                     91
C                    SETON                     92
C  N22               EXSR UPDATE
C   22               EXSR ENTER
C*-------------------------------------------------*
C* RESET BACK TO FIRST FORM AND REDO                 *
C*-------------------------------------------------*
C                    MOVEL"GDAR7"   FORMB
C                    MOVE "0"       RPTAPP
C                    MOVE "0"       FRZAPP
C                    EXSR INTFRM

C*-------------------------------------------------*
C* SPECIFY NEXT FORM NAME AND REPEAT/APPEND OPTIONS   *
C*-------------------------------------------------*
C                    EXSR CHGSUB
C*-------------------------------------------------*
C* RETURN TO START (AFTER HOUSECLEANING)             *
C*-------------------------------------------------*
C                    SETOF                     111213
C                    SETOF                     141516
C                    SETOF                     171920
C                    SETOF                     226791
C                    SETOF                     937475
C                    SETOF                     767778
C                    SETOF                     796692
C                    SETOF                     94
C                    GOTO START
C* END OF PROGRAM/PROCESSING
C          EXIT      TAG
C*************************************************
C*****          ENTER SUBROUTINE          ***********
C*************************************************
CSR        ENTER     BEGSR
C*-------------------------------------------------*
C* MOVE 2ND FORM NAME TO FORM NAME FIELD             *
C*-------------------------------------------------*
CSR        ENTST     TAG
CSR 92               MOVEL"GDAR8"   FORMB
CSR                  MOVE "0"       RPTAPP
CSR                  MOVE "0"       FRZAPP
CSR                  EXSR INTFRM
C*-------------------------------------------------*
C* SPECIFY NEXT FORM NAME AND REPEAT/APPEND OPTIONS   *
C*-------------------------------------------------*
```

```
CSR                     EXSR CHGSUB
CSR          ESTART     TAG
C*-------------------------------------------------------*
C* GET THE NEXT FORM FOR USE AND INITIALIZE THE FIELDS*
C* NOTICE THAT 92 IS SET ON. THIS IS USED WHEN CALLING*
C* ACTION "PUTDTA", AND DETERMINES WHICH SCREEN IS    *
C* BEING PROCESSED.                                   *
C*-------------------------------------------------------*
CSR                     EXSR INTSUB
C*-------------------------------------------------------*
C* DO GETDTA HERE TO CLEAR/RESET SCREEN               *
C*-------------------------------------------------------*
CSR                     EXSR GETSUB
C*-------------------------------------------------------*
C* MOVE 1ST SCREEN INFO TO 2ND SCREEN                 *
C*-------------------------------------------------------*
CSR 92                  MOVE BFREFN     SCREFN

C*-------------------------------------------------------*
C* REPLACE DATA IN V/3000 DATA BUFFER WITH DATA IN    *
C* USER PROGRAM BUFFER. (MOVE DATA FROM BUFFER TO SCN *
C*-------------------------------------------------------*
CSR                     EXSR PUTSUB
C* IF MASTER REC ADDED, DISPLAY MESSAGE
CSR          REPT       TAG
CSR                     SETOF                       111213
CSR                     SETOF                       141516
CSR                     SETOF                       171920
CSR                     SETOF                       7475
CSR                     SETOF                       767778
CSR                     SETOF                       79
C*-------------------------------------------------------*
C*SHOW FORM AND PLACE INPUT IN DATA BUFFER.            *
C*-------------------------------------------------------*
CSR                     EXSR SHWSUB
C*-------------------------------------------------------*
C* 6  * CLEAR MESSAGE WINDOW                          *
C*-------------------------------------------------------*
CSR                     EXSR CLRMSG
C*-------------------------------------------------------*
C* SEE IF FUNCTION KEY OR ENTER KEY IS HIT HERE. IF F2*
C* OR F8 NOT HIT, THE KEY IS INVALID AND A MSG IS SENT*
C* TO THE OPERATOR. IF ENTER, CONTINUE.               *
C*-------------------------------------------------------*
CSR 78                  GOTO ENDENT
CSR 77                  SETON                       76
CSR 75                  SETON                       76
CSR 76                  MOVE MSG1       MSG
CSR 76                  EXSR MSGD02
CSR 76                  GOTO REPT
C*-------------------------------------------------------*
C*  IF ENTER HIT - RUN VIEW EDITS                     *
C*-------------------------------------------------------*
CSR                     EXSR EDTSUB
C*** IF ERRORS - RETURN TO STEP 3
CSR 74                  GOTO REPT
C*** IF NO ERRORS, CONTINUE
C*-------------------------------------------------------*
C* TRANS DATA FROM DATA BUF TO UP BUFF, THEN READ REC*
C*-------------------------------------------------------*
CSR                     EXSR GETSUB
C*-------------------------------------------------------*
C* MOVE 2ND SCN INFO INTO HOLD AREA FOR DSPLY THE NEXT*
C* TIME THE FORM IS SHOWN.                            *
C*-------------------------------------------------------*
CSR 92                  EXSR HLDBF1
```

```
C*-------------------------------------------------*
C* REPLACE DATA IN V/3000 DATA BUFFER WITH DATA IN  *
C* USER PROGRAM BUFFER.                            *
C*-------------------------------------------------*
C                     EXSR PUTSUB
C*-------------------------------------------------*
C* ADD DETAIL LINE OF INVOICE HERE                 *
C*-------------------------------------------------*
CSR                   SETON                     81
CSR                   EXCPT
CSR                   SETOF                     81
C*-------------------------------------------------*
C* EXIT SUBROUTINE (AFTER HOUSECLEANING)           *
C*-------------------------------------------------*
CSR                   SETOF                     111213
CSR                   SETOF                     141516
CSR                   SETOF                     171920
CSR                   SETOF                     747576
CSR                   SETOF                     777879
C*-------------------------------------------------*
C* ROUTINE TO SWAP FORMS                           *
C*-------------------------------------------------*
CSR                   SETON                     91
CSR                   SETOF                     92
C*-------------------------------------------------*
C* END OF PROCESSING                               *
C*-------------------------------------------------*
CSR         ENDENT    TAG
CSR                   ENDSR

C*************************************************
C*******       UPDATE SUBROUTINE              *******
C*************************************************
CSR         UPDATE    BEGSR
C*-------------------------------------------------*
C* MOVE FORM NAME TO FORM NAME FIELD               *
C*-------------------------------------------------*
CSR         UPDST     TAG
CSR 92                MOVEL"GDAR8"    FORMB
CSR                   MOVE "0"        RPTAPP
CSR                   MOVE "0"        FRZAPP
CSR                   EXSR INTFRM
C*-------------------------------------------------*
C* SPECIFY NEXT FORM NAME AND REPEAT/APPEND OPTIONS *
C*-------------------------------------------------*
CSR                   EXSR CHGSUB
C*-------------------------------------------------*
C* GET THE NEXT FORM FOR USE AND INITIALIZE THE FIELDS*
C* NOTICE THAT 92 IS SET ON. THIS IS USED WHEN CALLING*
C* ACTION "PUTDTA", AND DETERMINES WHICH SCREEN IS  *
C* BEING PROCESSED.                                *
C*-------------------------------------------------*
CSR                   EXSR INTSUB
C*-------------------------------------------------*
C* DO GETDTA HERE TO CLEAR/RESET SCREEN            *
C*-------------------------------------------------*
CSR                   EXSR GETSUB
C*-------------------------------------------------*
C* GET INFO FROM DATA BUF TO USER PROG BUF (SCREEN) *
C*-------------------------------------------------*
CSR 92                EXSR MVSCN1
C*-------------------------------------------------*
C* REPLACE DATA IN V/3000 DATA BUFFER WITH DATA IN  *
C* USER PROGRAM BUFFER. (MOVE DATA FROM BUFFER TO SCN *
C*-------------------------------------------------*
CSR                   EXSR PUTSUB
C* 6US * DISPLAY FORM
```

```
CSR         REPD      TAG
CSR                   SETOF                    111213
CSR                   SETOF                    141516
CSR                   SETOF                    171920
CSR                   SETOF                    677475
CSR                   SETOF                    767778
CSR                   SETOF                    7966
C*-------------------------------------------------*
C*SHOW FORM AND PLACE INPUT IN DATA BUFFER.        *
C*-------------------------------------------------*
CSR                   EXSR SHWSUB

C*-------------------------------------------------*
C* 6  * CLEAR MESSAGE WINDOW                       *
C*-------------------------------------------------*
CSR                   EXSR CLRMSG
C*-------------------------------------------------*
C* SEE IF FUNCTION KEY OR ENTER KEY IS HIT HERE. IF F2*
C* OR F8 NOT HIT, THE KEY IS INVALID AND A MSG IS SENT*
C* TO THE OPERATOR. IF ENTER, CONTINUE.            *
C*-------------------------------------------------*
CSR 78                GOTO EXITSR
CSR 75                SETON                    66
CSR 77                SETON                    76
CSR 76                MOVE MSG1      MSG
CSR 76                EXSR MSGD02
CSR 76                GOTO REPD
C*-------------------------------------------------*
C*  IF ENTER HIT - RUN VIEW EDITS                  *
C*-------------------------------------------------*
CSR                   EXSR EDTSUB
C*** IF ERRORS - RETURN TO STEP 3
CSR 74                GOTO REPD
C*-------------------------------------------------*
C* TRANS DATA FROM DATA BUF TO UP BUFF, THEN READ REC*
C*-------------------------------------------------*
CSR                   EXSR GETSUB
C*-------------------------------------------------*
C* MOVE 2ND SCN INFO INTO HOLD AREA FOR DSPLY THE NEXT*
C* TIME THE FORM IS SHOWN.                         *
C*-------------------------------------------------*
CSR 92                EXSR HLDBG1
C*** SUPPLY USER EDITS HERE
C*-------------------------------------------------*
C* REPLACE DATA IN V/3000 DATA BUFFER WITH DATA IN  *
C* USER PROGRAM BUFFER.                            *
C*-------------------------------------------------*
CSR                   EXSR PUTSUB
C*-------------------------------------------------*
C* UPDATE RECORD WITH INFO FROM SCREEN TO FILE      *
C*-------------------------------------------------*
CSR                   SETON                    82
CSR                   EXCPT
CSR                   SETOF                    82
C*-------------------------------------------------*
C* EXIT SUBROUTINE (AFTER HOUSECLEANING)           *
C*-------------------------------------------------*
CSR                   SETOF                    111213
CSR                   SETOF                    141516
CSR                   SETOF                    171920
CSR                   SETOF                    677475
CSR                   SETOF                    767778
CSR                   SETOF                    797173
CSR                   SETOF                    66

C*-------------------------------------------------*
C* ROUTINE TO SWAP FORMS                          *
```

```
C*-------------------------------------------------------*
CSR                     SETON                       91
CSR                     SETOF                       92
C*-------------------------------------------------------*
C* END OF PROCESSING                                    *
C*-------------------------------------------------------*
CSR          EXITSR     TAG
CSR                     ENDSR
C*******************************************************
CSR          HLDBF1     BEGSR
CSR                     MOVE SCRCOD     BFRCOD  2
CSR                     MOVE SCREFN     BFREFN  8
CSR                     Z-ADDSCDATE     BFDATE  60
CSR                     MOVE SCITEM     BFITEM  8
CSR                     MOVE SCDESC     BFDESC 25
CSR                     Z-ADDSCCUST     BFCUST  60
CSR                     MOVE SCNAME     BFNAME 25
CSR                     Z-ADDSCTAX      BFTAX   33
CSR                     Z-ADDSCCOD      BFCOD   10
CSR                     Z-ADDSCTAXC     BFTAXC  10
CSR                     Z-ADDSCPAID     BFPAID  10
CSR                     Z-ADDSCCPST     BFCPST  10
CSR                     Z-ADDSCIPST     BFIPST  10
CSR                     MOVE SCBLNK     BFBLNK  5
CSR                     Z-ADDSCINVC     BFINVC  10
CSR                     Z-ADDSCDLOC     BFDLOC  20
CSR                     Z-ADDSCGEOC     BFGEOC  20
C* EDIT SCREEN TO FILE FIELDS
CSR                     MOVEASCQTY      CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT 10         QTYFLD  51
CSR                     Z-ADDQTYFLD     BFQTY   51
C* EDIT PRICE (SCREEN TO FILE) FIELD
CSR                     MOVEASCPRIC     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD  54
CSR                     Z-ADDPRCFLD     BFPRIC  54
C* EDIT COST (SCREEN TO FILE) FIELD
CSR                     MOVEASCCOST     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD
CSR                     Z-ADDPRCFLD     BFCOST  54
C* EDIT PRICE PER GALLON FIELD
CSR                     MOVEASCPRCG     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD
CSR                     Z-ADDPRCFLD     BFPRPG  54

C* MOVE MD AND FED PERCENTAGE TO EDIT BUFFER
CSR                     Z-ADDSCMPCT     BFMPCT  33
CSR                     Z-ADDSCFPCT     BFFPCT  33
C* EDIT EXTENSION FIELD
CSR                     MOVEASCEXTN     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BFEXTN  72
C* EDIT ROADTAX FIELD
CSR                     MOVEASCRTAX     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BFRTAX  72
C* EDIT SALETAX FIELD
CSR                     MOVEASCSTAX     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BFSTAX  52
C* EDIT TOTAL FIELD
CSR                     MOVEASCTOTL     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BFTOTL  72
```

```
CSR                     ENDSR
C*-------------------------------------------------*
CSR          HLDBG1     BEGSR
CSR                     MOVE SCRCOD     BGRCOD  2
CSR                     MOVE SCREFN     BGREFN  8
CSR                     Z-ADDSCDATE     BGDATE  60
CSR                     MOVE SCITEM     BGITEM  8
CSR                     MOVE SCDESC     BGDESC 25
CSR                     Z-ADDSCCUST     BGCUST  60
CSR                     MOVE SCNAME     BGNAME 25
CSR                     Z-ADDSCTAX      BGTAX   33
CSR                     Z-ADDSCCOD      BGCOD   10
CSR                     Z-ADDSCTAXC     BGTAXC  10
CSR                     Z-ADDSCPAID     BGPAID  10
CSR                     Z-ADDSCCPST     BGCPST  10
CSR                     Z-ADDSCIPST     BGIPST  10
CSR                     MOVE SCBLNK     BGBLNK  5
CSR                     Z-ADDSCINVC     BGINVC  10
CSR                     Z-ADDSCDLOC     BGDLOC  20
CSR                     Z-ADDSCGEOC     BGGEOC  20
C* EDIT SCREEN TO FILE FIELDS
CSR                     MOVEASCQTY      CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT 10         QTYFLD  51
CSR                     Z-ADDQTYFLD     BGQTY   51
C* EDIT PRICE (SCREEN TO FILE) FIELD
CSR                     MOVEASCPRIC     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD  54
CSR                     Z-ADDPRCFLD     BGPRIC  54

C* EDIT COST (SCREEN TO FILE) FIELD
CSR                     MOVEASCCOST     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD
CSR                     Z-ADDPRCFLD     BGCOST  54
C* EDIT PRICE PER GALLON FIELD
CSR                     MOVEASCPRCG     CARA
CSR                     EXSR NMEDIT
CSR          EDTFLD     MULT .01        PRCFLD
CSR                     Z-ADDPRCFLD     BGPRPG  54
C* MOVE MD AND FED PERCENTAGE TO EDIT BUFFER
CSR                     Z-ADDSCMPCT     BGMPCT  33
CSR                     Z-ADDSCFPCT     BGFPCT  33
C* EDIT EXTENSION FIELD
CSR                     MOVEASCEXTN     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BGEXTN  72
C* EDIT ROADTAX FIELD
CSR                     MOVEASCRTAX     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BGRTAX  72
C* EDIT SALETAX FIELD
CSR                     MOVEASCSTAX     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BGSTAX  52
C* EDIT TOTAL FIELD
CSR                     MOVEASCTOTL     CARA
CSR                     EXSR NMEDIT
CSR                     Z-ADDEDTFLD     BGTOTL  72
CSR                     ENDSR
C*-------------------------------------------------*
CSR          MVSCN1     BEGSR
CSR                     MOVE GERCDE     SCRCOD
CSR                     MOVE GEREFN     SCREFN
CSR                     Z-ADDGEDATE     SCDATE
CSR                     MOVE GEITEM     SCITEM
```

```
CSR                     MOVE GEDESC     SCDESC
CSR                     Z-ADDGECUST     SCCUST
CSR                     MOVE GENAME     SCNAME
CSR                     Z-ADDGETAXP     SCTAX
CSR                     Z-ADDGECOD      SCCOD
CSR                     Z-ADDGETAXC     SCTAXC
CSR                     Z-ADDGEPAID     SCPAID
CSR                     Z-ADDGECPST     SCCPST
CSR                     Z-ADDGEIPST     SCIPST
CSR                     MOVE BLANK1     SCBLNK
CSR                     Z-ADDGEINVC     SCINVC
CSR                     Z-ADDGEDLOC     SCDLOC
CSR                     Z-ADDGEOC       SCGEOC

C* EDIT QTY FILE TO SCREEN FIELD
CSR                     SETON                       70
CSR           GEQTY     MULT .1         FLDQTY  72
CSR                     Z-ADDFLDQTY     TESTFD  72
CSR                     EXSR EDITSB
CSR                     MOVE EARA       SCQTY
C* EDIT PRICE (FILE TO SCREEN) FIELD
CSR                     SETON                       69
CSR           GEPRIC    MULT 100        FLDPRC  52
CSR                     Z-ADDFLDPRC     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE EARA       SCPRIC
C* EDIT COST (FILE TO SCREEN) FIELD
CSR                     SETON                       69
CSR           GEUCST    MULT 100        FLDPRC
CSR                     Z-ADDFLDPRC     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE EARA       SCCOST
C* EDIT PRICE PER GALLON (FILE TO SCREEN) FIELD
CSR                     SETON                       69
CSR           GEPRPG    MULT 100        FLDPRC
CSR                     Z-ADDFLDPRC     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE EARA       SCPRCG
C* MOVE MD AND FED PERCENTAGE TO EDIT BUFFER
CSR                     Z-ADDGEMPCT     SCMPCT
CSR                     Z-ADDGEFPCT     SCFPCT
C* EDIT EXTENSION FIELD
CSR                     Z-ADDGEEXTN     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE DARA       SCEXTN
C* EDIT ROADTAX (FILE TO SCREEN) FIELD
CSR                     Z-ADDGERTAX     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE DARA       SCRTAX
C* EDIT SALETAX FIELD
CSR                     SETON                       68
CSR                     Z-ADDGESTAX     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE EARA       SCSTAX
C* EDIT TOTAL (FILE TO SCREEN) FIELD
CSR                     Z-ADDGETOTL     TESTFD
CSR                     EXSR EDITSB
CSR                     MOVE DARA       SCTOTL
CSR                     ENDSR

C*****************************************************************
C**********          CLEAR MESSAGE WINDOW          ***********
C*****************************************************************
CSR           CLRMSG    BEGSR
CSR                     MOVE "60"       MSLEN   2
CSR                     MOVE "J"        ENHCD
CSR                     MOVE MSG2       MSG
```

```
CSR                     SETON                          50
CSR                     EXCPT
CSR                     SETOF                          50
CSR                     MOVEL"SHOW   "  ACTION
CSR                     SETON                          55
CSR                     EXCPT
CSR                     SETOF                          55
CSR                     ENDSR
C*****************************************************
C* THIS SUBROUTINE EDITS NUMBERS FROM FILE TO SCREEN
CSR           EDITSB    BEGSR
CSR                     MOVEAMSG2       CARA
CSR                     MOVEAMSG2       ARA
CSR                     MOVEAMSG2       ARA1
CSR                     MOVEAMSG2       ARA2
CSR           TESTFD    COMP 0                             61
CSR 61        TESTFD    MULT 2          TESTF1   72
CSR 61        TESTFD    SUB  TESTF1     TSTFD1   72
CSRN61                  Z-ADDTESTFD     TSTFD1
CSR                     MOVE TSTFD1     TEST1    7
CSR                     MOVEATEST1      ARA
CSR 61                  MOVE "-"        CARA,1
CSRN61                  MOVE " "        CARA,1
CSR                     Z-ADD0          X
CSR           LOOP01    TAG
CSR           X         ADD  1          X
CSR           ARA,X     COMP "0"                           60
CSR 60                  MOVE " "        ARA,X
CSR           X         COMP 7                             62
CSR 62                  GOTO ENDL01
CSR 60                  GOTO LOOP01
CSR           ENDL01    TAG
CSR                     Z-ADD0          X
CSR                     Z-ADD1          Y
C* ARA,1 THRU ARA,5 MOVED TO CARA,2 THRU CARA,6.
CSR           LOOP02    TAG
CSR           X         ADD  1          X
CSR           Y         ADD  1          Y
CSR                     MOVE ARA,X      CARA,Y
CSR           X         COMP 5                             63
CSRN63                  GOTO LOOP02
CSR                     MOVE "."        CARA,7
CSR                     MOVE ARA,6      CARA,8
CSR                     MOVE ARA,7      CARA,9

C* GO TO 5 OR 6 DIGIT EDITS OR EXIT THIS SUBROUTINE
CSR 70                  GOTO SUBR1
CSR 69                  GOTO SUBR2
CSR 68                  GOTO SUBR3
CSR 67                  GOTO SUBR4
CSR                     GOTO ENDSR1
C* ROUTINE TO CONVERT 5.1 FIELDS
CSR           SUBR1     TAG
CSR                     MOVE CARA,9     ARA1,7
CSR                     MOVE "."        ARA1,6
CSR                     MOVE CARA,8     ARA1,5
CSR                     MOVE CARA,6     ARA1,4
CSR                     MOVE CARA,5     ARA1,3
CSR                     MOVE CARA,4     ARA1,2
CSR                     MOVE CARA,1     ARA1,1
CSR                     GOTO ENDSR1
C* ROUTINE TO CONVERT 5.4 FIELDS
CSR           SUBR2     TAG
CSR                     MOVE CARA,9     ARA1,7
CSR                     MOVE CARA,8     ARA1,6
CSR                     MOVE CARA,6     ARA1,5
CSR                     MOVE CARA,5     ARA1,4
```

```
CSR                     MOVE "."        ARA1,3
CSR                     MOVE CARA,4     ARA1,2
CSR                     MOVE CARA,1     ARA1,1
CSR                     GOTO ENDSR1
C* ROUTINE TO CONVERT 5.2 FIELDS
CSR          SUBR3      TAG
CSR                     MOVE CARA,9     ARA1,7
CSR                     MOVE CARA,8     ARA1,6
CSR                     MOVE CARA,7     ARA1,5
CSR                     MOVE CARA,6     ARA1,4
CSR                     MOVE CARA,5     ARA1,3
CSR                     MOVE CARA,4     ARA1,2
CSR                     MOVE CARA,1     ARA1,1
CSR                     GOTO ENDSR1
C* ROUTINE TO CONVERT 6.0 FIELDS
CSR          SUBR4      TAG
CSR                     MOVE CARA,9     ARA2,6
CSR                     MOVE CARA,8     ARA2,5
CSR                     MOVE CARA,6     ARA2,4
CSR                     MOVE CARA,5     ARA2,3
CSR                     MOVE CARA,4     ARA2,2
CSR                     MOVE CARA,3     ARA2,1
C* END OF EDITS
CSR          ENDSR1     TAG
CSR                     MOVEACARA       DARA    9
CSR                     MOVEAARA1       EARA    7
CSR                     MOVEAARA2       FARA    6
CSR                     MOVEAMSG2       ARA
CSR                     MOVEAMSG2       CARA
CSR                     MOVEAMSG2       ARA1
CSR                     MOVEAMSG2       ARA2
CSR                     SETOF                           606162
CSR                     SETOF                           636768
CSR                     SETOF                           6970
CSR                     ENDSR

C*****************************************************
C* THIS SUBROUTINE EDITS NUMBERS FROM SCREEN TO FILE.
CSR          NMEDIT     BEGSR
CSR                     MOVEAMSG2       ARA
CSR                     MOVEAMSG2       ARA1
CSR                     MOVEAMSG2       ARA2
CSR                     Z-ADD1          X       20
CSR                     Z-ADD1          Y       20
CSR          CARA,1     COMP "-"                        60
CSR 60                  MOVE " "        CARA,1
CSR                     Z-ADD9          X
CSR          LOOP3      TAG
CSR          X          COMP 0                          62
CSR 62                  GOTO ENDL03
CSR          CARA,X     COMP " "                        61
CSR 61       X          SUB  1          X
CSR 61                  GOTO LOOP3
CSR          ENDL03     TAG
C* IF FIELD IS BLANK (X=0), RESET INDEX X TO 9
CSR          X          COMP 0                          65
CSR 65                  Z-ADD9          X
CSR                     Z-ADD7          Y
CSR          LOOP4      TAG
CSR          X          COMP 0                          63
CSRN63       Y          COMP 0                          63
CSR 63                  GOTO ENDL04
CSR          CARA,X     COMP "."                        64
CSR 64       X          SUB  1          X
CSR 64                  GOTO LOOP4
CSR                     MOVE CARA,X     ARA,Y
CSR          X          SUB  1          X
```

```
CSR          Y          SUB  1          Y
CSR                     GOTO LOOP4
CSR          ENDL04     TAG
CSR                     MOVEAARA        TEST02  7
CSR                     MOVE TEST02     TEST03  70
C* QTY EDITED (70 ON), NO DECIMALS, SO Z-ADD TO EDIT1
CSR          TEST03     MULT .01        EDIT1   72
CSR 60       EDIT1      MULT -1         EDTFLD  72
CSRN60       EDIT1      MULT 1          EDTFLD
CSR                     MOVEAMSG2       CARA
CSR                     MOVEAMSG2       ARA
CSR                     SETOF                           606162
CSR                     SETOF                           636465
CSR                     ENDSR
C*-------------------------------------------------*
C*V/3000 SCREEN SUBROUTINES
C*-------------------------------------------------*
CSR          INTFRM     BEGSR
CSR                     MOVE "          "NXTFRM 15
CSR                     MOVEL"          "NXTFRM
CSR                     MOVELFORMB      NXTFRM
CSR                     ENDSR
C*-------------------------------------------------*
C*SUBROUTINE TO CHANGE FORMS-B
CSR          CHGSUB     BEGSR
CSR                     SETON                       58    CHGNXT-50
CSR                     EXCPT
CSR                     SETOF                       58
CSR                     ENDSR

C*-------------------------------------------------*
C*SUBROUTINE TO GET NEXT FORM AND INITIALIZE-C
CSR          INTSUB     BEGSR
CSR                     MOVEL"GETNXT"  ACTION  6
CSR                     SETON                       55    GETNXT-51
CSR                     EXCPT
CSR                     SETOF                       55
C* SET INITIAL VALUES
CSR                     MOVEL"INIT   "  ACTION
CSR                     SETON                       55    INIT-58
CSR                     EXCPT
CSR                     SETOF                       55
CSR                     ENDSR
C*-------------------------------------------------*
C*SHOW THE FORM AND PLACE INPUT IN V/3000 DATA BUF-D
CSR          SHWSUB     BEGSR
CSR                     MOVEL"SHOW   "  ACTION
CSR                     SETON                       55    SHOW-53
CSR                     EXCPT
CSR                     SETOF                       55
C* READ FROM TERMINAL
CSR                     MOVEL"RDTERM"  ACTION
CSR                     SETON                       55    RDTERM-54
CSR                     EXCPT
CSR                     SETOF                       55
CSR                     READ TRANSFIL                   H0
CSR                     ENDSR
C*-------------------------------------------------*
C*SUBROUTINE TO PERFORM USER EDITS IF ENTER HIT-E
CSR          EDTSUB     BEGSR
CSR                     MOVEL"EDITS "  ACTION
CSR                     SETON                       55    EDITS-59
CSR                     EXCPT
CSR                     SETOF                       55
CSR                     READ TRANSFIL                   H0
CSR                     ENDSR
C*-------------------------------------------------*
```

```
C*SUBROUTINE TO GET DATA BUF INFO TO USER PROG BUF-F
CSR          GETSUB    BEGSR
CSR                    MOVEL"GETDTA"  ACTION
CSR                    SETON                        55    GETDTA-64
CSR                    EXCPT
CSR                    SETOF                        55
CSR                    READ TRANSFIL                    H0
CSR                    ENDSR
C*-------------------------------------------------*
C*SUBROUTINE TO PUT USER PROG BUF INFO TO DATA BUF-G
CSR          PUTSUB    BEGSR
CSR                    SETON                        57    PUTDTA-63
CSR                    EXCPT
CSR                    SETOF                        57
CSR                    ENDSR

C*-------------------------------------------------*
C*SUBROUTINE TO PUT USER PROG BUF INFO TO DATA BUF-G
CSR          SHDSUB    BEGSR
CSR                    SETON                        59    SHODTA-57
CSR                    EXCPT
CSR                    SETOF                        59
CSR                    ENDSR
C*-------------------------------------------------*
CSR          MSGD01    BEGSR
CSR                    MOVE "60"       MSLEN
CSR                    MOVE "J"        ENHCD
CSR                    MOVE MSG1       MSG
CSR                    SETON                        50
CSR                    EXCPT
CSR                    SETOF                        50
CSR                    ENDSR
C*-------------------------------------------------*
CSR          MSGD02    BEGSR
CSR                    MOVE "60"       MSLEN   2
CSR                    MOVE "J"        ENHCD   1
CSR                    SETON                        50
CSR                    EXCPT
CSR                    SETOF                        50
CSR                    ENDSR
C*-------------------------------------------------*
CSR          MSGD03    BEGSR
CSR                    MOVE "60"       MSLEN   2
CSR                    MOVE "J"        ENHCD   1
CSR                    SETON                        51
CSR                    EXCPT
CSR                    SETOF                        51
CSR                    ENDSR
C*-------------------------------------------------*
CSR          MSGD04    BEGSR
CSR                    MOVE "60"       MSLEN   2
CSR                    MOVE "J"        ENHCD   1
CSR                    SETON                        52
CSR                    EXCPT
CSR                    SETOF                        52
CSR                    ENDSR
C*-------------------------------------------------*

O*** INDICATOR 55 FOR ACTIONS 51,53,54, 58-61, 64-70
O*** INDICATOR 56 FOR ACTIONS 56 AND 62
O*** INDICATOR 57 FOR ACTIONS 57 AND 63
O*** INDICATOR 58 FOR ACTION 50
OTRANSFILE         55
O                             ACTION     6
O          E       56
O                                        6 "CORERR"
O                             FLDNO     11
```

```
0                          MSLEN     13
0                          ENHCD     14
0                          MSG       75
0         E        50
0                                     6 "PUTMSG"
0                          MSLEN      8
0                          ENHCD      9
0                          MSG       69
0         E        52
0                                     6 "SHOMSG"
0                          MSLEN      8
0                          ENHCD      9
0                          MSG       69
0         E        51
0                                     6 "BADFLD"
0                          FLDNO     11
0                          MSLEN     13
0                          ENHCD     14
0                          MSG       75
0* THIS IS THE PUTDTA FOR ENTRY DATA.....
0         E        57 91
0                                     6 "PUTDTA"
0                          DATALN    10
0                          SAREFN    18
0         E        57 92
0                                     6 "PUTDTA"
0                          DATALN    10
0                          SCRCOD    12
0                          SCREFN    20
0                          SCDATE    26
0                          SCITEM    34
0                          SCDESC    59
0                          SCCUST    65
0                          SCNAME    90
0                          SCTAX     93
0                          SCCOD     94
0                          SCTAXC    95
0                          SCPAID    96
0                          SCCPST    97
0                          SCIPST    98
0                          SCBLNK   103
0                          SCINVC   104
0                          SCDLOC   106
0                          SCGEOC   108
0                          SCQTY    115
0                          SCPRIC   122
0                          SCCOST   129
0                          SCPRCG   136
0                          SCMPCT   139

0                          SCFPCT   142
0                          SCEXTN   151
0                          SCRTAX   160
0                          SCSTAX   167
0                          SCTOTL   176
0         E        58
0                                     6 "CHGNXT"
0                          NXTFRM    21
0                          RPTAPP    22
0                          FRZAPP    23
0* SETON 82 ONLY IF IT IS TO UPDATE FILE; IF NOT SETOF 82
OGDENTER EADD      81
0                          BFRCOD     2
0                          BFREFN    10
0                          BFDATE    16
0                          BFITEM    24
0                          BFDESC    49
```

```
0                       BFCUST    55
0                       BFNAME    80
0                       BFTAX     82P
0                       BFCOD     83
0                       BFTAXC    84
0                       BFPAID    85
0                       BFCPST    86
0                       BFIPST    87
0                       BFBLNK    92
0                       BFINVC    93
0                       BFDLOC    95
0                       BFGEOC    97
0                       BFQTY     100P
0                       BFPRIC    103P
0                       BFCOST    106P
0                       BFPRPG    109P
0                       BFMPCT    111P
0                       BFFPCT    113P
0                       BFEXTN    117P
0                       BFRTAX    121P
0                       BFSTAX    124P
0                       BFTOTL    128P
0       EDEL    82 66
0                       BGRCOD     2
0                       BGREFN    10
0                       BGDATE    16
0                       BGITEM    24
0                       BGDESC    49
0                       BGCUST    55
0                       BGNAME    80
0                       BGTAX     82P
0                       BGCOD     83
0                       BGTAXC    84

0                       BGPAID    85
0                       BGCPST    86
0                       BGIPST    87
0                       BGBLNK    92
0                       BGINVC    93
0                       BGDLOC    95
0                       BGGEOC    97
0                       BGQTY     100P
0                       BGPRIC    103P
0                       BGCOST    106P
0                       BGPRPG    109P
0                       BGMPCT    111P
0                       BGFPCT    113P
0                       BGEXTN    117P
0                       BGRTAX    121P
0                       BGSTAX    124P
0                       BGTOTL    128P
0       E       82
0                       BGRCOD     2
0                       BGREFN    10
0                       BGDATE    16
0                       BGITEM    24
0                       BGDESC    49
0                       BGCUST    55
0                       BGNAME    80
0                       BGTAX     82P
0                       BGCOD     83
0                       BGTAXC    84
0                       BGPAID    85
0                       BGCPST    86
0                       BGIPST    87
0                       BGBLNK    92
0                       BGINVC    93
```

```
      0                          BGDLOC    95
      0                          BGGEOC    97
      0                          BGQTY    100P
      0                          BGPRIC   103P
      0                          BGCOST   106P
      0                          BGPRPG   109P
      0                          BGMPCT   111P
      0                          BGFPCT   113P
      0                          BGEXTN   117P
      0                          BGRTAX   121P
      0                          BGSTAX   124P
      0                          BGTOTL   128P
**       MSG1
FUNCTION KEY INVALID AT THIS TIME-TRY AGAIN.
**       MSG2  (ALL BLANKS FOR CLEARING MESSAGE WINDOW)

**       MSG3

MASTER RECORD ADDED - HIT ENTER KEY TO CONTINUE
**       MSG4
RECORD IS NOT IN FILE-HIT F1(ADD) OR RETRY
**       MSG5
NO MORE INVOICE LINES AVAILABLE-REKEY #
**       MSG6
UPDATE RECORD NOT IN FILE-HIT F8 TO EXIT UPDATE SUBR
**       MSG7
MASTER RECORD NOT IN FILE-EXIT SUBROUTINE
**       MSG8
ID# NOT IN GDID FILE.
**       MSG9
NO MORE PMT LINES ON THIS CR MEMO-KEY IN NEW NUMBER
**       MSG10
ID # IS BLANK-KEY DESCRIPTION ON NEXT SCREEN
**       MSG11
DELIVERY LOCATION IS BLANK-KEY IN LOCATION
**       MSG12
QTY X PRICE NE EXTN-PRESS ENTER TO CONTINUE
**       MSG13
SALES/ROAD TAX SHOULD NOT BE ON SAME TICKET
**       MSG14
TOTAL NOT CORRECT-PRESS ENTER TO CONTINUE
```

Biographical Sketch of Christopher J. Colburn G.D. Armstrong Company, Inc. Laytonsville, Maryland

Christopher J. Colburn, is a programmer/analyst presently employed by the G.D. Armstrong Company of Laytonsville, Maryland. He "got his start" in programming while working as an operator/programmer with business applications (in RPG) on an IBM System 34 at the Leisure World of Maryland Corporation from 1978 to 1981. In May of 1980, he received a Bachelor of Science degree in Business Administration from the University of Maryland. From 1981 to 1982, he worked at Computer Applications and Systems, Inc., a local computer consulting firm. There he created new and modified existing programs in Accounts Payable, Accounts Receivable, and Job-costing business applications on the HP3000, using both RPG and COBOL languages.

In January of 1983, he accepted his present position of programmer/analyst with G.D. Armstrong Company, Inc., a fuel oil and automotive accessory distributor. His accomplishments at Armstrong include: conversion of existing Payroll and A/R programs from an IBM System 32 to the HP3000, creation of an automotive parts inventory system, and development of an A/P-Inventory-A/R database system for use in Armstrong's day-to-day business operations.

PLAY IT AGAIN, KSAM

GARY TODOROFF
DATAMASTER COMPUTER SERVICE

INTRODUCTION

Keys have long been a simple method of organizing and accessing data. For instance, a library's card catalog gives multiple key access by author, title and subject; in the phone book, the white pages are keyed alphabtically by name, and the yellow pages, by product or service. Keyed Sequential Access Method (KSAM) uses the same simple approach based on the concept of keys. Perhaps that's why KSAM feels natuaral and familiar to many programmers.

Of course, real world situations do not always lend themselves to simplicity, especially in data processing. Sometimes data structures and their relationships are very complex. But if certain criteria are met, KSAM is the logical choice for a data processing application. Instead of adding to the complexity, KSAM may simplify the approach with easy, direct retrieval of information. I like to think that KSAM stands for Keep Simple Access Methods.

KSAM HISTORY

A brief historical note will help to set the stage as well as clear up some misconceptions. Perhaps initially, in response to the marketing opportunity and demand for compatability with IBM's System/3 minicomputer, KSAM was rather hastily assembled to replace an even earlier effort called RSAM. KSAM was a bit shakey at first.

Problems were often encountered, such as end-of-file pointers not agreeing between MPE and KSAM. Guess-the-actual-number-of--records in the KSAM file became a not-so-popular game played by many programmers. Even more serious, KSAM files could be left in an unpredictable state after system failures, with nobody the wiser. Folks accustomed to the rock-solid dependability of ISAM (Indexed Sequential Access Method) on the IBM systems of the 1970's encountered some rude suprises with the early KSAM.

However, except for one notorious "fix" to KSAM (more later), conditions have improved greatly. The original VIEW/3000 (now VPLUS) used KSAM for its form files, providing HP good incentive for a major overhaul of KSAM back in MPE release 1918. KSAMUTIL, the HP utility used to build, study, and fix KSAM files also received new features to help solve some of the old problems. Most notable was the "crash flag" (which will let KSAM automatically recover and correct a file damaged by a system failure.) File problems are also quickly fixed with the KEYINFO command in KSAMUTIL which will resolve any key-file problems. File locking was another early improvement to KSAM, enhanced even more when dynamic locking became available to RPG with the "LOCK" and "UNLOCK" operations in a more recent release.

Overall, the improvements have been needed and appreciated. But one "feature" which overwrites KSAM records must be mercilessly exposed! During all those past improvements, KSAM was supposedly modified to be more consistent with IBM ISAM, specifically update operations to a KSAM file when no record was retrieved prior to a file update. For example, consider a batch file which is used to update a KSAM master file. If the batch file is empty, a record in the KSAM file may be overlayed with blanks. Apparently, unless special precautions are taken, the KSAM file is rudely updated while the logical file pointer is still relaxing at the first record awaiting where to go next. With no input batch records and no place better to go, WHAM, there go your empty input batch record buffers, splattering

whatever used to be in the first record of the KSAM file.

Especially with RPG programs, first records can be overwritten. The condition has existed for three years now, causing incredible amounts of confusion, file repairs, reprogramming, and possibly even causing undetected file errors produced by programs which use to run predictably and flawlessly. (Figure 1 shows a sample RPG program in which the problem could occur and the fix required). Known Problem Reports (KPRs) have been submitted to HP concerning the problem many times. So far, the only acknowledgement is a warning on compiler listings of programs which could encounter this situation. Until HP fixes this problem, be forewarned!

(Late news from HP indicates that the above problem may be solved by allowing "Update-Protect-Checking" (UPC) in RPG version 5.08 to be relased with MPE V, S-MIT. UPC is enabled by placing a "U" in column 28 of the RPG Header specification. Using this new feature, indicators would not need to be set on in the calculations as shown in the RPG program of Figure 1.)

KSAM STRENGTHS

Disclaimers now aside, let's proceed to the practical use of KSAM by outlining its strengths for data base functions.

- Both keyed access and chronological access may be used with KSAM files. While the key-file maintains the index, records are added sequentially, resulting in a sometimes useful chronological record order.

- KSAM files may be separated into file extents, thereby not claiming disproportionate amounts of disc space when not full of data. (IMAGE allocates all space immediately and even backs up the blank records to tape.)

- KSAM files may be filled right up to End-of-File without sacrificing performance, again resulting in more effecient disc usage.

- KSAM files do not require special utilities for loading and unloading data. Simple FCOPY commands may be used after a KSAM file is built. In fact, FCOPY itself will build a KSAM file of the same characteristics as the original KSAM file through a very simple use of parenthesis in the FCOPY command. For example:

```
FCOPY FROM=KSAMFILE;TO=(DATAFILE,KEYFILE)
```

FCOPY will create a new file called "DATAFILE" which will be identical to and have the same key-file structure as the original file, "KSAMFILE".

- KSAM key fields may be updated directly.

- Deleted data is retreivable. Since only the key is removed from the key-file, the data file still contains the deleted records which may be retreived later if necessary. Again, using the FCOPY example, deleted records may be recovered using:

```
FCOPY FROM=KSAMFILE;TO=NEWFILE;NEW;NOKSAM
```

\column 2

The first two bytes of the deleted data record were overlayed when the record was deleted, but otherwise the record is still intact.

- Keys may be defined as unique or with duplicate key values allowed. One recent feature to possibly increase performance with duplicates is the "RDUP" parameter of the KSAMUTIL BUILD command. RDUP allows for duplicate keys without maintaining the chronoligical order in which duplicate key records were added to the key-file.

- Standard MPE file handling may be used in most cases with KSAM files. For example, file equations allow a KSAM file to be opened with ";ACC=OUT", thereby easily setting the file pointer to the beginning of the file, deleting any previous data. Even the EDITOR may be used to make occaisional or global changes to a KSAM file, for example:

```
:EDITOR
/TEXT KSAMFILE,UNN
```

```
/CHANGEQ "PARTNO  " TO "PART-NUM" IN ALL
/KEEP $NEWPASS,UNN
/EXIT
:FCOPY FROM=$OLDPASS;TO=KSAMFILE
```

- Perhaps one of KSAM's biggest advantages is to allow a file to be processed sequentially by key. Especially if listings by key are often required either on paper or the terminal, this provides a real advantage over the multi-step process with IMAGE, which may involve extracting data from a data set, sorting it and finally listing it. Some users have stored just key information in KSAM files, with IMAGE used for storing the actual data, providing an external sequential by key access to IMAGE data bases.

- An easy method of jumping into a KSAM file is allowed with partial key or generic key access. From that point in the file, records may then be processed sequentially by key. Figure 2 provides a sample of how to do this with RPG.

KSAM DISADVANTAGES

Before mentioning some special aspects of KSAM, a few of the disadvantages should be mentioned, too.

- HP does not provide a QUERY-like language for accessing KSAM files.

- Except for standard file lock words, KSAM does not allow for file security at the record and/or field level.

- Keys must be contiguous. Multiple keys may overlap, but cannot start in the same position.

- Without careful locking strategies, concurrent record access may cause problems.

- Interrelations in data sets are not automatically handled by KSAM , but must be done within the application software.

- KSAM opens an Extra Data Segment for each file and user, which can increase resource usage and require special techniques for file sharing.

Of course, one usually finds a way around most disadvantages, or at least some way to excuse them. I won't excuse any, but will mention some of the software and techniques available to overcome some of KSAM's limitations.

KSAM UTILITY PROGRAMS

KSMQUERY from the Contributed Library provides a "quick and dirty" access to KSAM files. Primarily intended as a file debugging tool, KSMQUERY does open up files very simply for inspection and extracts records easily by key values of equal to, less than or greater than. If for example, invoice number is a key, and you want to see all records since invoice number 1000, the KSMQUERY syntax looks like this:

```
>FILE INVOICES      (specify file name)
>RECS 50            (display up to 50 records)
>SEARCH GT          (show records with keys greater than)
>=1000              (start search at key value, 1000)
```

KSMQUERY will then display records to terminal or line printer.

KSMQUERY provides an additional capability which is unique on the HP3000. By using the "DUMP HEX" command, packed fields are very easily read with the four bit "halves" of each byte displayed in decimal form below each character. Until I either lose two fingers or gain six, octal and hexidecimal will never feel comfortable, so it's a real pleasure with KSMQUERY to read packed fields directly. (See example in Figure 3) In fact, a couple of times I've even used FCOPY to put a file subset

into a quickly built KSAM file just to use the HEX display feature, a very handy way to read packed fields.

Another useful program from the Contributed Library is KSAMRBLD. KSAM files can be modified by answering interactive prompts for blocking factor, file limit, and number of extents. It will even pick out an effecient blocking factor for you, then rebuild the file with your new sizes.

KSAM DEFAULT FEATURES

One of the advantages of KSAM is the way it defaults to workable file structures if you don't provide the parameters. Obscure areas such as key blocking factors will be taken care of automatically. In fact, the best approach is often not to think too hard about all the various options. For example, the FIRSTREC option of the KSAMUTIL BUILD command allows you to override the normal first record equals zero default. (To begin counting at "zero" rather than "one" never ceases to amaze me; people have counted starting with the number one since recorded history. Then some computer engineer decided to start counting with zero. Future archeologists will no doubt puzzle over this strange practice.) Nevertheless, use the KSAM default of zero for consistency and everything works fine.

MULTIPLE KEY USE

KSAM allows for an amazing sixteen possible keys per record--most of my files don't have that many fields. I have always been curious to try out so much horsepower, just to see what happens. Actually, two keys is the most I've ever used, which keeps me well within the usual warnings to avoid using much more than three keys per file. Of course, more keys could be used, especially for files that are primarily for inquiry, with updates being done only rarely or where speed and disc I/O are not important, such as adding records in batch mode to an archival file that is updated monthly. Adding records with multiple keys can cause a large amount of reshuffling the key-file, so consider the available resources carefully.

One of those occaisonal two-key files I have used provides an interesting example of the KSAM key features. A file of job records needed to be accessed both by part-number/job-number(a key made up of the two fields) and by part-number only, but in date order. The original sequential file was sorted by date within part-number before copying it into a KSAM file. The job-number was not used as a sort field, even though it will become part of the KSAM key. But note that both keys involved the part-number. Keys may overlap but not start in the same position. To solve that, a blank character was always placed to the left of the part-number itself so that one of the keys could start in that position. The record layout was something like this:

```
 column
position..  3   4       8                32
        /BLANK/PART-NO/JOB-NO/...data.../DATE/
          ^^^^^^^^^^^^^^
          key #1 ^
                 ^^^^^^^^^^^^^
                      key#2 ^
```

The various steps to create this file are as follows:

```
:RUN SORT.PUB.SYS
INPUT JOBMPE
OUTPUT JOBSORT
KEY 3,4          (the part-no field)
KEY 32,6         (the date field in YYMMDD format)
END

:RUN KSAMUTIL.PUB.SYS
BUILD DJOB;REC=-80,16,F,ASCII;DISC=150000,15;&
KEYFILE=KJOB;KEY=B,3,5,,DUP;KEY=B,4,8,,DUP
EXIT

:FCOPY FROM=JOBSORT;TO=DJOB
```

Access by key #2 is rather straight-forward, since records can be processed by part-number and job-number keyed sequence. Key #1 provides a more interesting access. First, duplicate keys are used. Since the DUP parameter (as opposed to RDUP) was used in the KSAMUTIL BUILD command, the chronological order of the key file was maintained as the records were added to the KSAM file. Therefore whenever records are accessed by key #1, they are always in date order, (since the original MPE file used date as a sort field) even though the date field is not part of any key. Also note that by starting key #1 with a blank character one byte to the left of the part-number, we overcame the limitation which does not allow KSAM multiple keys to start in the same position. Note too that key #1 starts in postion three of the record, leaving the first two bytes empty since KSAM uses that word as the delete flag.

Still on the same file example, an interesting mistake occurred when loading this file which contained around 150,000 records. Due to a typo, the sort was not done by the part-number key field as planned. Thus the "sorted" file was in random order from the standpoint of the key. The file took over six hours to load. After the problem was discovered, the file was resorted correctly, and FCOPY accomplished the load in about 50 minutes. (The computer was a very lightly loaded Series 44) Draw your own conclusions on how to order files before batch adds or updates to KSAM files.

SETLL COMMAND FOR KSAM FILES IN RPG

One final example using the above file involves a simple coding technique in RPG demonstrating how to sequentially process the file beginning with a selected part-number. The RPG calculation operation, SETLL (Set Lower Limit) is used with the KSAM file defined as Input Demand. Records are read and output for as long as the part-number matches the one selected. (See Figure 2)

KSAM FILE SHARING AND LOCKING

Straying farther from the Keep Simple Access Methods philosophy, let's discuss file sharing and locking. Again the examples will be given in RPG. One of KSAM's disadvantages concerns the use of Extra Data Segments (XDS) for each file that is opened by each user. Since a global control for data is not available, some special techniques need to be used to insure data integrity.

In a shared file environment, one file accessor has a personal "snap-shot" of data and key-structure stored in one XDS per user. The danger is that a record may be retrieved and changed while a second user has completed the same, resulting in a modified key-structure which no longer corresponds to the XDS information of our first user. When the first user then updates the file, the wrong record could be updated, since record pointers may no longer be valid. (To explain fully would require a lengthly discussion of the KSAM "B-tree" used for indexing files. References are given in the bibliography, since internal KSAM structures are beyond the scope of this article.)

Locking not only needs to be employed, but must also be done with the right technique to insure that the current and accurate copy of the key-file is in the XDS as the time the file update is done. Primarily, this requires that the file is read once to get a data record, then read once more to re-position the file pointer in case any changes were made to the key-file by another user. By locking around both operations, data will be updated properly. Optionally, the file could simply be locked for the entire inquiry and update operation. The file would be unavailable to other users while one user has a record displayed on the terminal. This technique has drawbacks and could cause definite problems amongst users unless, of course, they all take their coffee breaks together.

If extensive file sharing will be done with KSAM, then please read HP's Communicator Number 21. Otherwise the simple solution is to just avoid file situations that require KSAM file locking and sharing.

FILE STRUCTURE AND BLOCKING

Blocking factors for KSAM data files may be approached in the same way as MPE files, that is, try to use disc space effeciently by blocking on sector boundaries. However, disc utilization is not the only factor, especially if the KSAM file will most often be accessed randomly. A large blocking factor for the sake of disc usage could create a waste of memory by requiring a large XDS to contain the data block. In a large file, the chances are remote that the next

randomly accessed record will be in the same data block already in memory; large blocking factores will not improve disc I/O. Keep data blocking factors small for files that are usually accessed randomly.

If, however, the file will be processed sequentially most often, then consider the average number of records that make up a "unit" of data, for example the number of entries per customer invoice. If that number is 15, then a blocking factor of approximately that size would be appropriate to avoid extra disc I/O if displaying an invoice for that customer. As usual, the trade-offs are between disc and memory usage, and should be judged by the type of file being used.

Blocking factors also apply to KSAM key-files. Once again, I must defer to another article which explains in detail how significant performance improvements may be obtained through the use of effecient blocking factors for key-files. I strongly reccommend that you obtain Jorge Guerrero's article on KSAM Design Guidelines. His paper provides one of the clearest explanations available of the B-tree structure, and explains how to use blocking factors to keep B-tree levels to a minimum, thereby decreasing disc I/O very significantly. The tables included in Guerrero's article go way beyond anything provided in the KSAM manual, yet are simple to use for optimizing key-file blocking factors. If you are at all concerned with KSAM file performance, those key-blocking tables are a must. (See Bibliography)

For further information, Appendix B of the KSAM manual is well worth reading. File structures are covered in depth.

CONCLUSION

My favorite part in the KSAM manual is the first sentence on page B-1, which provides a good point for concluding:

> "KSAM files can be used efficiently without any knowledge of how the files are structured or how file blocking and size is determined."

In other words, you don't have to be an expert to use KSAM on the HP3000. The simple approach of using keys to retrieve data provides information in a way as easy to understand as the yellow pages. KSAM files are easy to create and maintain. They are also relatively effecient in their use of disc space. Various utility routines let you manipulate KSAM files, and the ease of defining and reading KSAM files with RPG gives an especially high-level approach to programming. If the best approach can be defined as the simplest approach, then don't allow things like data structures to become complicated in the first place. For keeping simple access methods, KSAM is a good place to start.

FIGURE 1. "Work around potential corruption of first record in KSAM update file by conditioning the update with an indicator set on in calculation specifications."

```
     H
     H* A BATCH FILE UPDATES A KSAM FILE.  BE SURE TO SETON
     H* INDICATOR 90 IN CALCULATIONS TO PREVENT CORRUPTING THE
     H* FIRST RECORD OF THE KSAM FILE IF THE BATCH FILE IS EMPTY
     H*

     F* BATCH INPUT FILE
     FBATCH   IP  F  80  80            DISC
     F* KSAM UPDATE-ADD FILE (COL-15="U" AND COL-66="A")
     FMASTER  UC  F 128 128R 6AI     3 DISC

     IBATCH   NS  01
     I                                        1   6 INKEY L1
     I*... etc
     IMASTER  NS  02
     I                                        3   8 KSKEY
     I*... etc

     C* SETON INDICATOR FIRST TIME CALCULATIONS ARE DONE
     C*
     C  N90                SETON                    90
     C*
     C* CHAIN TO MASTER FILE TO BE UPDATED
     C*
     CL1          INKEY     CHAINMASTER              66
     C*... etc

     O* UPDATE MASTER FILE AT LEVEL BREAK L1 WHEN INKEY CHANGES
     OMASTER  T       N66 L1 90
     O*...output fields, etc
```

Note that the output is conditioned by indicator 90 which will only be on if there were records in the BATCH file. If the output was not conditioned by 90 and the BATCH file was empty, the first record in the KSAM file would be overlaid with blank field buffers, since indicator 66 is not on and the L1 level indicator is set on along with the last record LR indicator.

FIGURE 2. "The RPG Set Lower Limit (SETLL) calculation allows a simple way to process a KSAM file sequentially within a defined limit."

```
     H
     H* THIS PROGRAM READS A KSAM FILE SEQUENTIALLY WITHIN
     H* LIMITS USING THE "SETLL" CALCULATION.
     H*

     F* KSAM INPUT-DEMAND FILE (COL-15="I" AND COL-16="D")
     F*
     FJOBFILE ID  F  80  80L 5AI     3 DISC
     F*                   ^ ^        ^ KEY STARTS COLUMN THREE
     F*                   ^ ^ KEY IS FIVE BYTES LONG
     F*                   ^ PROCESS BY LIMIT
     F*
     F* OUTPUT FILE DEFINED ON LINE PRINTER
     F*
     FOUTFILE O   F 132 132            LP

     IJOBFILE NS  01
     I                                        3   7 PART#
     I                                        3   5 PART3
     I*... etc

     C* ACCESS TO KSAM IS "GENERIC" BY FIRST THREE BYTES
     C* OF THE PART NUMBER.  SET UP A PARTIAL KEY TO
     C* USE AS KSAM FILE POINTER
     C*
     C                   MOVE "123  "   KEY05   5
     C           KEY05   SETLLJOBFILE
     C*
     C* DO READ LOOP UNTIL FIRST THREE BYTES OF KEY DO NOT MATCH
     C*
     C           RLOOP   TAG
     C                   READ JOBFILE                  60 E-O-F
     C  N60      PART3   COMP "123"              60     >
     C  N60              EXCPT
     C  N60              GOTO RLOOP
     C*... etc

     O* OUTPUT IS TO PRINTER BASED ON "EXCPT" WHILE KEY IS
     O* EQUAL TO "123" IN FIRST THREE POSITIONS.
     O*
     OOUTFILE E       N60 01
     O*...output fields, etc
```

FIGURE 3. "KSMQUERY allows packed decimal fields to be read easily by displaying each half of the byte in decimal format."

Type the HEX command in KSMQUERY before displaying a record. The record will display on the terminal 40 bytes at a time with column indicators.

In the example below, "^^^^" denotes a birthdate field in bytes 35 to 38 of the record. The field is stored in packed decimal format. By reading the bottom two rows diaganolly, starting with the lower zero, the date can be read as "071256". The "q%l" on the first row is "garbage" that comes from interpreting the field as ASCII.

```
                1...5....0....5....0....5....0....5....0....5
1 ...   40  00001805SMITH          HENRY   KM.q%10
           2333333335445422222222224445522244072642
           0000018053D948000000000085E29000BD015CF0
                                              ^^^^
```

BIBLIOGRAPHY

HP COMMUNICATOR, NUMBER 21, MPE 1918, "TIPS ON INSURING KSAM FILE INTEGRITY", P. 13

FOLKINS, DALE, HEWLETT PACKARD, "KSAM - IT'S ALIVE AND WELL" CONFERENCE PROCEEDINGS, P. L-9

GUERRERO, JORGE, HEWLETT PACKARD, "KSAM DESIGN GUIDELINES FOR OPTIMIZATION", IUG COPENHAGEN PROCEEDINGS, 1982

KAMINSKI, STEVE, "KSAM VS. IMAGE", JOURNAL OF THE HP GENERAL SYSTEM USERS GROUP, VOL 1 NO 6, MARCH/APRIL 1978, PP. 16-18

As General Manager of Datamaster Computer Service, Gary Todoroff has been serving a diversified group of clients since 1975. Along with consulting in systems management and programming, he also directs five other programmers in applications on several HP3000 computers and a few micro and IBM computers. Specialties include IBM to HP conversions, programmer productivity tools, and applicatons in health-care, wholesale distribution, direct mail, and fund-raising. Most recently, he has been especially involved with designing and marketing ORBIX Control Language, an application supervisor system which also allows IBM System/34/36 software to run directly on the HP3000.

Based in Eureka, California between San Francisco and Portland, Gary is involved with both the Bay Area (BAYRUG) and Northwest (NOWRUG) Regional User Groups. He started HUMBUG, Humboldt County's own user group and is a supporter of RPGSIG, which he hopes will encourage use of RPG and other high-level language use on the HP3000. Gary may be reached in Eureka at 707/445-8425.

BURN BEFORE READING - HP3000 SECURITY AND YOU.

by Eugene Volokh,
VESOFT, INC.

DISCLAIMER

One of the most important things you can do for your security system is to plug holes that may exist in it. To help you do this, this paper shows some ways in which most inadequately secured systems can be penetrated. Although this information can be used by would-be security violators to break into a poorly secured system, it is in my opinion more important that it can be used by you to protect yourself against these violators.

ATTENTION WOULD-BE THIEVES! DO NOT READ ANY FURTHER!

INTRODUCTION

Life's not fair.

Just when you think you've got it made, just when you (or your company) have found the pot of gold at the end of the rainbow, you find it out.

Someone else wants the same pot.

To prevent your property from becoming theirs, you set up a series of obstacles between the would-be thief and your property. It is these obstacles that comprise your security system, and it is the quality of these obstacles that determines whether or not your property (in our case, your computer data) is secure.

These obstacles come in two flavors:

* Obstacles to unauthorized retrieval of data. Data is often valuable in and of itself, whether it is salary information you want to keep secret from your employees, financial information you want to keep secret from your competition, or military information you want to keep secret from THEM.

* Obstacles to unauthorized modification of data. Data does not exist for its own sake; real-life decisions are made based on that data, and unauthorized modification of data can affect those decisions in an undesirable way.

This paper will try to give you some useful tips on making your valuable data more secure.

THE ROAD TO YOUR DATA

Consider Joe Q. Sinister, who has his sights set on your payroll database. There is a fixed road that he must travel to reach the data stored in it and change it; knowing this road will help us erect the proper roadblocks.

His first step must be to log on to the computer; if we can frustrate his attempts to do that, our data is secure. The techniques used to prevent unauthorized users from logging on to the computer are called LOGON SECURITY.

LOGON SECURITY

Logon security is probably the most important component in your security fence. This is because many of the further security devices (e.g. file security) use information that is established at logon time, such as user id and account name. Thus, we must not only forbid unauthorized users from logging on, but must also ensure that

even an authorized user can only log on to his user id.

So, logon security essentially involves ensuring that the person logging on is authorized to use the user id he is logging on to. How is this to be done?

The optimal approach, of course, would be to somehow identify who the person is (fingerprints? retina scan?) and check to see if he is on the authorization list for the particular user id. Unfortunately, these approaches are not within the means of most HP 3000 users. However, another good method is.

A person can be identified by what he knows almost as well as by what he looks like. For instance, a user id may be assigned a password, and only the people authorized to use that user id may be told that password. Then (assuming no one else somehow learns the password), if a person knows the password, it follows that he is authorized. Alternatively, if one and only one user is allowed to use a particular user id, he may be asked to enter some personal information (mother's maiden name?) when he is initially added to the system, and then be asked that question (or one of a number of such personal questions) every time he logs on. This general method of determining a user's authorizations by what he knows we will call "knowledge security".

Unfortunately, the knowledge security approach, although one of the best available has one major flaw -- unlike fingerprints, information is easily transferred, be it revealed voluntarily or involuntarily; thus, someone who is not authorized to use a particular user id may nonetheless find out the user's password. You may say: "Well, we change the passwords every month, so that's not a problem". The very fact that you have to change the passwords every month means that they tend to get out through the grapevine! A good security system does not need to be redone every month, especially since that would mean that at least towards the end of the month, the system is already rather shaky and subject to penetration.

Ironically, the biggest culprit in this respect is the user himself. Users have been often known to write down passwords and post them in prominent places so they will not forget them; reveal passwords to people who really shouldn't know them; and, in general, wreak havoc on your logon security system. Some ways have been designed to cope with this, such as the personal profile security system (asking questions such as "What's your mother's maiden name?", "Where did you go on your first date?", etc.) described above, whose main advantage is that users are less likely to reveal personal data than impersonal passwords; additionally, there can be more than one personal profile password -- all of them or a random one can be asked at logon time -- whereas there is only user password. However, the user is still the weakest link in the logon security system, and major steps should be taken to avoid voluntary password disclosure by the user. Thus, an important security rule arises:

* THE USER IS THE WEAKEST LINK IN THE LOGON SECURITY SYSTEM -- DISCOURAGE HIM FROM REVEALING PASSWORDS (by techniques such as personal profile security or even by reprimanding people who reveal passwords -- they seem innocent, but they can lose you millions).

Yet another way in which passwords are often revealed is by having job streams with embedded passwords. First of all, unless you take special precautions (such as altering the job streams so that Read access to them is disallowed, and only Execute -- enough for :STREAMing -- is permitted), anyone who can stream the job stream can also read it and thus see the passwords; in any case, any listing of the job stream (of which plenty are liable to be laying around the computer room) contains this password. More importantly, since changing a password means having to change every single job stream that contains it, these passwords are virtually guaranteed never to be changed. Fortunately, there is a simple way to resolve this problem: there are plenty of programs, contributed and vendor-supported, that take a job stream without embedded passwords, prompt for them, insert them into the job stream, and then stream it.

* PASSWORDS EMBEDDED IN JOB STREAMS ARE EASY TO SEE AND VIRTUALLY IMPOSSIBLE TO CHANGE -- AVOID THEM.

Another way of increasing logon security is by indirectly using another aspect of user identification -- identification by human beings. Actually, this could be the main part of your logon security system: any user who wishes to sign on must first get clearance from a security guard or console operator. Going quite this far is too expensive, but a little bit of this can be obtained for free.

If some 15-year-old high school student walks into your data entry area and starts using the computer, people are bound to notice. It is fear of being identified as a security violator by other human beings that makes most violation attempts come across phone lines, usually at night or on weekends. Thus, another useful security feature is to be able to restrict access by access location (i.e. terminal) and access time. The very fact that someone is trying to run payroll across a phone line at 11 PM on a Saturday is an indication of unauthorized access. Thus, it is worthwile to implement some form of security that prohibits access to certain user id's and accounts at certain times of day, days of week, and/or from certain terminals. Alternatively, you might want to force people to answer an additional password at certain times, or especially when signing on from certain terminals.

This may seem like a poor approach indeed -- after all, if the thief hits the time of day, day of week, or terminal prohibition/password, this means that he has successfully penetrated the rest of your security system, which will never happen -- right? In reality, this is a very potent way of frustrating would-be security violators, especially if the attempted violators are promptly investigated. Thus, another maxim appears:

* SOME FORMS OF ACCESS ARE INHERENTLY SUSPECT (AND THUS REQUIRE EXTRA PASSWORDS) OR ARE INHERENTLY SECURITY VIOLATIONS. THUS, ACCESS TO CERTAIN USER ID'S AT CERTAIN TIMES OF DAY, ON CERTAIN DAYS OF WEEK, AND/OR FROM CERTAIN TERMINALS (SUCH AS DIAL-IN OR DS LINES) SHOULD BE SPECIALLY RESTRICTED.

ASIDE -- ATTEMPTED VIOLATION REPORTING

Before we go any further with our discussion of various security devices, it is worthwhile to pay particularly close attention to something which should be present in all security devices -- violation reporting.

No security system can cover you 100% -- given enough time, a determined (or even relatively casual) thief can penetrate even the best system. Fortunately, before this one successful penetration, chances are that the thief will make many unsuccessful attempts; if you pay attention to these unsuccessful attempts, you can catch the thief (or at least improve the security system by, say, temporarily shutting down dial-in lines) before he gets in.

This may seem obvious, but few shops really pay attention to unsuccessful penetration attempts -- when was the last time you looked at "INVALID PASSWORD" messages on the system console or in the log files? In reality, every incorrect password entry is indication of a possible attempted security violation, even more so if there are several such errors in a row.

HP doesn't help any either -- the INVALID PASSWORD messages look just like any other console message (no enhancements of any kind); the only place where invalid password entries are logged are in the system log files together with the rest of the console log messages. It would have been far more desirable if the message were logged to a separate log file, and maybe even reported to the line printer or some special device. Additionaly, it might be wise for a terminal on which an invalid password entry occurs to be shut down for some period of time so that it would take more time for a would-be thief to try more passwords.

But, even with the existing HP system, an alert console operator can nip many a potential security violation in the bud by catching the INVALID PASSWORD messages that can be a sign of an attempted violation. In fact, there is a way to highlight some messages so they will be more easily visible. Since most MPE messages are stored in the system file called CATALOG.PUB.SYS, you can do the following:

1. Sign on as MANAGER.SYS.

2. In EDITOR (or TDP), /TEXT CATALOG.PUB.SYS

3. Modify the first line in the file that starts with "65 " and the first line that starts with "68 " to contain an escape sequence such as "escape&dB" (inverse video) right after the blank after the message number and to contain a "escape&d@" (turn off enhancement) at the end of the message. Alternatively, if you have a 263x with expanded character set, insert an "escape&k1S" (enter expanded set) right after the blank after the message number and a "escape&k0S" (exit expanded set) at the end of the message. Similar escape sequences may be put in if you have some other kind of terminal or a voice output device.

4. /KEEP the file as INPUT.

5. :RUN MAKECAT.PUB.SYS,BUILD

6. Presto! Your "INVALID PASSWORD" and "MISSING PASSWORD" messages are now much easier to read.

Thus:

* MANY SECURITY VIOLATIONS CAN BE AVERTED BY MONITORING THE WARNINGS OF UNSUCCESSFUL VIOLATION ATTEMPTS THAT OFTEN PRECEDE A SUCCESSFUL ATTEMPT. IF POSSIBLE, CHANGE THE USUAL MPE CONSOLE MESSAGES SO THEY WILL BE MORE VISIBLE.

LOGOFF SECURITY

Another threat to your system security is, unfortunately, a rather common one. If someone signs on to a terminal and then walks away (for a lunch break, say), a would-be thief can access your computer without even having to log on -- just walk up to the terminal and use it.

You may think this to be a relatively rare occurence, but consider: do your people always sign off when they go to lunch? Haven't there been times when they forgot to sign off even when they leave for the day? Leaving a terminal signed on is a very common mistake, and one that can greatly jeopardize the security of your system.

How can you solve this problem? Well, for one, you can tell your people -- whenever they leave the terminal, they should sign off. Alternatively, if you find that people often leave the terminal when it's in some particular state (say, the main menu of your accounts payable program), set a timeout just before issuing the terminal read (with the FCONTROL intrinsic, mode 4). That way, when the user does not respond for a certain amount of time, the read will abort, and your program will be able to terminate and maybe log off the user. An even better alternative is to use a contributed or vendor-supplied program that automatically aborts all terminals that have been inactive for more than a certain amount of time (such as Boeing's BOUNCER or VESOFT's LOGOFF).

Another, more dangerous, problem occurs when a dial-in user hangs up the phone instead of properly :BYEing off. Then, if the dial-in line is configured with subtype 0, the user will not be automatically :BYEd off, and the next person to call up the computer will be dropped into the still-logged in session. Thus, remember to configure all your dial-in lines with subtype 1 or tell your users in no uncertain terms that they MUST always :BYE off when using the dial-in line.

Thus,

* LEAVING A TERMINAL LOGGED ON AND UNATTENDED IS JUST AS MUCH A SECURITY VIOLATION AS REVEALING THE LOGON PASSWORD. USE SOME KIND OF TIMEOUT FACILITY TO ENSURE THAT TERMINALS DON'T REMAIN INACTIVE FOR LONG; SET UP ALL YOUR DIAL-IN TERMINALS WITH SUBTYPE 1.

RESTRICTED VS. UNRESTRICTED USER INTERFACE

As was mentioned before, logon security is a very important component of your security system, but it is by no means the only one. Many security violations are committed by people who are allowed to sign on to the computer but manage to get at things that they are not permitted to access.

There are two major ways of forbidding authorized users from doing unauthorized things. One is by permitting them to do only certain specific things (the inclusive approach) and the other is by forbidding them from doing specific things (the exclusive approach). Each has its merits, its uses, and its security strategies.

THE INCLUSIVE APPROACH

Briefly, the inclusive approach is usually implemented by having an OPTION LOGON, NOBREAK (the NOBREAK is important!) UDC that runs an application program and then, upon exit from the program, immediately BYEs. Thus, the user is only allowed to perform the function or functions of this one program (or, if the program so wishes, only a subset of these functions), and he is forbidden from doing anything else -- accessing files, running programs, or executing MPE commands.

This is, overall, a good approach. Its only real problem is that in some instances, it is too restrictive -- some users (especially programmers) need to have access to the

entire power of MPE. However, when the user does not need to access MPE, it is not only more secure but also more convenient for the user to be automatically dropped into his program when he signs on and be automatically signed off when he exits the program. However, certain technical issues must be kept in mind:

1. Don't forget to make the UDC OPTION LOGON, NOBREAK. If you omit the NOBREAK, the user can hit break, type :ABORT, and get into MPE.

2. A less-known fact is that it is usually essential that you add a CONTINUE line before running your program, thus making your UDC look something like

   LOGONUDC OPTION LOGON, NOBREAK CONTINUE RUN ACCPAY.PUB.AP BYE

   Why? Because otherwise, if the program aborts the entire UDC will be flushed and the BYE will never be encountered. Although it might seem quite improbable that your program will abort, the user can actually make most programs abort by typing a :EOD (or sometimes just a :) when prompted for input. This causes an end of file on $STDIN and makes many programs, including almost all BASIC, COBOL, FORTRAN, and PASCAL programs, abort.

Of course, this approach need not be restricted to running simple applications programs. One of the best uses of this approach is to run some program that displays a menu of allowed MPE commands or constructs and asks the user to choose one. Thus, if you want a user to access the A/P system, EDITOR, or the TELLOP command, you might write a program that displays these three options to the user, asks the user for one, and then executes it (via the COMMAND or CREATE intrinsic). Even better, get a general-purpose menu procesing program that permits you to easily set up various menus by just changing some data files. Thus,

* A USEFUL APPROACH TO SECURING YOUR SYSTEM IS TO SET UP A LOGON MENU WHICH ALLOWS THE USER TO CHOOSE ONE OF SEVERAL OPTIONS RATHER THAN TO LET THE USER ACCESS MPE AND ALL ITS POWER DIRECTLY.

THE EXCLUSIVE APPROACH

Sometimes, programmers or other users that have to use a wide range of programs, files, and MPE commands must have access to MPE itself. This is a far less controlled environment than a program that is run at logon time, but can still be secured very well.

One approach to securing the system while still allowing people to access MPE is to disable certain MPE commands you find undesirable. For instance, say you do not want your people to :STREAM jobs. You could set up a system or account UDC

```
STREAM !FILENAME="$STDIN", !COLON="!"
OPTION LIST
COMMENT  YOU ARE NOT ALLOWED TO :STREAM FILES.
```

That way, whenever someone types a :STREAM command, he gets the UDC instead.

This approach, however, has a major flaw: Although the command interpreter gives precedence to UDCs over ordinary MPE commands (thus allowing you to block out :STREAM commands by setting up a STREAM UDC), the COMMAND intrinsic does not. Thus, if the user is allowed to access FCOPY, EDITOR, TDP, SPOOK, or even a user-written program that calls the COMMAND intrinsic, he will be able to bypass the UDC restriction. In other words, in the example above, all I need do to bypass the :STREAM command restriction is to run FCOPY, and type the :STREAM command from there!

The only exceptions to the above rule are the commands that can not be directly executed via the COMMAND intrinsic, such as :RUN, :PREP, the compiler commands, :SETCATALOG, and :SHOWCATALOG. But even these commands (all except :SETCATALOG and :SHOWCATALOG) are available through some programs, such as TDP and SPOOK.

Thus,

* BLOCKING OUT MPE COMMANDS VIA UDC'S WITH THE SAME NAME WILL USUALLY FAIL UNLESS THE COMMAND IS :SETCATALOG OR :SHOWCATALOG OR IF YOU ALSO FORBID ACCESS TO MANY HP SUBSYSTEMS AND HP-SUPPLIED PROGRAMS. THIS SEVERELY LIMITS THE USEFULNESS OF THIS METHOD.

Again, I'd like to stress that the :SETCATALOG and :SHOWCATALOG can be blocked out this way, as can (with more difficulty) the :RUN command and some other commands; however, the set of commands still permitted will usually be so small, the method involved so complex, and the chance of penetration so great, that all advantages of the exclusive approach pale in comparison.

By far the best way, in my opinion, of implementing the exclusive approach is by using the existing MPE file, database, and program security features, which is what the next few sections will discuss.

FILE SECURITY

File security is quite possibly the most sophisticated and the least used and understood security system provided by MPE. If properly handled, it can permit a user to use all MPE commands and all of MPE's power without allowing him to go beyond the confines of his files.

Each file has a so-called "security matrix", an array of information that describes what classes of users can read, write, append, execute, and/or lock a file. Similarly, each group has a security matrix describing the security to be set for its files, and each account also has a security matrix. These security matrices are the things that

LISTDIR2 shows you when you do a LISTSEC (or LISTF, LISTGROUP, or LISTACCT).

When a user tries to open a file, MPE checks to see if the user is allowed to access the file by the account security matrix, by the group security matrix, and the file security matrix. If he is allowed by all three, the file is opened; if at least one security matrix forbids access by this user, the open fails. For instance, if we try to open TESTFILE.JOHN.DEV when logged on to an account other than DEV and the security matrix of the group JOHN.DEV forbids access by users of other accounts, the open will fail (even though both TESTFILE's and DEV's security matrices permit access by users of other accounts).

Each security matrix describes which of the following classes can READ, WRITE, EXECUTE, APPEND to, and LOCK the file:

* CR - File's creator
* GU - Any user logged on to the same group as the file is in
* GL - User logged on to the same group as the file is in and having Group Librarian (GL) capability
* AC - Any user logged on to the same account as the file is in
* AL - User logged on to the same account as the file is in and having Account Librarian (AL) capability
* ANY - any user
* Any combination of the above (including none of the above)

By default, whenever any account is created, access to all its files is restricted to AC (account users only), except for the SYS account, in which Read and Execute is allowed for ANY and Write, Append, and Lock for AC; whenever any group is created, access to all its files is restricted to GU (group users only), except if the group is PUB, in which case access is Read and Execute for AC (all account users) and Write, Append, and Lock for GU (group users) and AL (account librarian); and whenever any file is created, access to it is allowed to everyone. Incidentally, a System Manager can access (in any mode) any file in the system, and an Account Manager any file in his account.

Thus, let us say that you, who build your files in JOHN.DEV, wish other users to be able to read your files. To do this, you have to go to your account manager, get him to allow Read access to the group JOHN.DEV for ANY, and get him to ask the system manager to allow Read access to DEV for ANY. This, needless to say, is rather complicated, and, in fact, most users go the much easier route of just :RELEASEing their files.

However, the problem with :RELEASEing a file is that when you do it, ANYBODY is

allowed to do ANYTHING to the file -- this means read it, write to it, even purge it! And, since doing this is so easy, many files are :RELEASEd and never re-:SECUREd, thus leaving them open for easy tampering by anyone; another contributing factor to this is that ordinary MPE :LISTF does not show whether or not the file has been :RELEASEd, so many people don't even know which of their files are :RELEASEd.

However, if getting the access restrictions on your group and account loosened is so difficult, but :RELEASEing the file makes it wide-open for any kind of access, what is to be done? Unfortunately, the solution is by no means easy.

The first step is to set up all your accounts with all forms of access allowed to ANY; i.e. alter them with a command such as

```
:ALTACCT
accountname;ACCESS=(R,W,A,L,X:ANY)
```

This still leaves a level of security (group security) that will by default protect the file (except for PUB groups, which should thus be built with Read and Execute access for AC instead of ANY), while making the security much easier to waive -- one would need to lift group security only instead of group and account security.

Next, when building each group, consider closely the security that you would wish to put on it. If, for instance, this group consists mostly of files that should be readable by anybody, build it with Read access allowed to ANY. Files can then be protected individually by :ALTSECing them to a more restrictive security level.

Finally, if you :RELEASE a file so that someone can access it, be sure to :SECURE it immediately after the other person is done (unless you don't care about security for that file). It's even better if you have some global file manipulation utility (such as VESOFT's MPEX) with which you can :SECURE all the files in some fileset that have been :RELEASEd.

Thus, some important file security guidelines exist:

* REMEMBER THAT :RELEASE'ING A FILE LEAVES IT WIDE OPEN FOR ANY KIND OF ACCESS; :RELEASE FILES CAUTIOUSLY, AND RE-:SECURE THEM AS SOON AS POSSIBLE.

* TRY TO MAKE IT AS EASY AS POSSIBLE FOR PEOPLE TO ALLOW THEIR FILES TO BE ACCESSED BY OTHERS WITHOUT HAVING TO :RELEASE THEM. THUS, BUILD ALL ACCOUNTS WITH (R,W,X,A,L:ANY) SO THAT ALLOWING ACCESS TO A GROUP WILL BE EASIER.

* IF A GROUP IS MOSTLY COMPOSED OF FILES THAT SHOULD BE ACCESSIBLE BY ALL USERS IN THE SYSTEM OR BY ALL ACCOUNT USERS, BUILD IT THAT WAY. THIS WILL ALSO REDUCE :RELEASE'S.

* THE :ALTSEC COMMAND IS USEFUL FOR RESTRICTING ACCESS TO FILES IN A GROUP TO WHICH ACCESS IS NORMALLY LESS RESTRICTED.

One more aspect of file security that bears mentioning is the file lockword. With it, you could conceivably restrict access to a file to only those users (or programs!) who know the file lockword, even if the file's security matrix says that they have complete access to the file. However, the problem with lockwords is the same as the problem with passwords -- they don't stay secret for long. In my opinion, other security approaches (better use of the security matrices, user id checks in programs being protected, etc.) are superior.

* LOCKWORDS AREN'T ALL THEY'RE CRACKED UP TO BE. OTHER APPROACHES SHOULD BE PREFERRED.

ASIDE -- ALLOWING PROGRAMS TO READ :SECURE'D FILES

Say that you want your accounts payable program to ask the user for a password and then check the user's input against a password stored in a file. Now, you naturally can't store the password in a :RELEASEd file, for then the password would be readable by anybody; however, if it is stored in a :SECUREd file, then the program won't be able to access it, either, since the program is run by ordinary users.

One solution is to :RELEASE a file, but put a lockword on it. Then, the program could open the file specifying a lockword, but users will not be able to open a file because they won't know the lockword. This is a relatively good solution; however, its flaw is that, like all passwords, the lockword is likely to become known sooner or later. Then, the entire advantage of storing the password in a file, namely that the password can be easily changed,

will be nullified by the fact that the file's lockword can not be easily changed.

A different approach uses an undocumented feature of the FOPEN intrinsic. If FOPEN is called in privileged mode, and the 4 low-order bits of the "aoptions" parameter (third from the left) are set to 15, the file is opened for read access IGNORING ALL SECURITY. This is not a security violation because it requires PM capability (see the CAPABILITIES section); however, since PM need only be granted the program and the group and account in which it resides (which could be PUB.SYS), the program will be able to access the file regardless of who is running it, but most users will not (since the file can thus be :SECUREd).

CAPABILITIES

There are some MPE capabilities that have a bearing on system security.

Of these, SM and AM are simple to explain and relatively well understood -- they allow one to access (in any way) all files in the system, and the account, respectively.

Some others -- AL and GL -- allow one to establish classes of users (Librarians) that are allowed to access files that other users may not because they can be explicitly allowed access by the security matrices (see FILE SECURITY).

However, the security effects of two other capabilities -- OP and PM -- are often not properly appreciated, much to the detriment of system security.

OP CAPABILITY

OP capability, which stands for System Supervisor (NOT Operator!), has one capability that has a very large bearing on system security: a user with OP capability can :STORE and :RESTORE any file on the system. This might not mean much, but this really means that

A USER WITH OP CAPABILITY CAN READ AND WRITE ANY FILE IN THE SYSTEM

After all, to read it, all he has to do is to :STORE it and then FCOPY the tape to the line printer; and to write to it, he can store it, move it to a system on which he has write access to the file's group and account, modify it, store it again, and restore it on the original system. Can you trust your operators (who are usually given this capability) with this kind of power?

* YOU SHOULD ONLY GIVE OP CAPABILITY TO USERS WHO YOU TRUST AS MUCH AS YOU WOULD A SYSTEM MANAGER, TO USERS WHO HAVE NO ACCESS TO MAGNETIC TAPES OR SERIAL DISCS, OR TO USERS WHO HAVE A LOGON UDC THAT DROPS THEM INTO A MENU WHICH FORBIDS THEM FROM DOING :STORE'S OR :RESTORE'S

PM CAPABILITY

No capability has been feared, discussed, or maligned quite as much as PM capability. In this paper, I will only discuss the the security ramifications of PM capability; for a discussion of PM and system crashes, see my paper "Privileged Mode: Use and Abuse".

What does PM capability give you? Quite simply, it allows you to obtain SM capability as follows:

```
:DEBUG
?MDL-'DL-1'+2
DL-NNN     MMMMM := -1
?E
```

Once you do this, you are (at least partially) a system manager until you log off. You can access any file and even execute system manager commands like :ALTACCT and :ALTGROUP to give yourself SM or any other capability permanently.

Obviously, PM capability is not something you want to give to every Tom, Dick, and Harry.

* YOU SHOULD ONLY GIVE PM CAPABILITY TO USERS WHO YOU TRUST AS MUCH AS YOU WOULD A SYSTEM MANAGER.

However, there are other ways in which users can get PM capability.

For one, for a program to have PM capability (and thus use various privileged operating system functions), the program must reside in a group and account which have PM capability. This is very good -- that way, programs like DBUTIL and SPOOK, which use privileged mode, can be run by plain vanilla users who do not have to be given PM. However, this means that if a privileged program does something to circumvent normal MPE security (see the ASIDE -- ALLOWING PROGRAMS TO READ :SECURE'D FILES), it'll do it for anybody who runs

it, unless it explicitly checks who is running it.

More importantly, this means that a user does not need to have PM capability to write privileged programs -- only the ability to build files in a privileged group (i.e. S [Save] access to that group) or to overwrite a program file in that group with his own file (i.e. W [Write] access to any program file in that group) and then run them (i.e. X access to the program file being overwritten or any access if he has S access to the group -- then he can just release the file).

For instance, say that I work out of EUGENE.DEV and the group PROG.DEV has PM capability and Save access for all account users. I can just write a program that uses privileged mode to access a file that I shouldn't be able to access or to grant myself all the capabilities (like in the :DEBUG example above), :PREP it without CAP=PM (since :PREPping with CAP=PM requires PM capability), then change the program file to have PM capability (a task that does not require privileged mode), and copy it into PROG.DEV. Although while the program was in EUGENE.DEV, I couldn't run it (since it is required that the group in which the program is have PM capability), once it is in PROG.DEV, I could run it. If I don't have execute access to PROG.DEV, I can :RELEASE the program before running it, since I'm the creator of the file.

Or, say that somebody :RELEASEd any program file in PUB.SYS, thus giving me write and execute access to it. Then, I can write a program that uses privileged mode to bypass system security, :PREP it without CAP=PM, change the program file to have PM capability, and copy it on top of that program file in PUB.SYS. Then, since PUB.SYS has PM capability and I have execute access to the file I just overwrote, I can run the program.

Thus,

* IF ANY USER HAS SAVE ACCESS TO A GROUP WITH PM CAPABILITY, OR WRITE AND EXECUTE ACCESS TO ANY PROGRAM FILE THAT RESIDES IN A GROUP WITH PM CAPABILITY, HE CAN WRITE AND RUN PRIVILEGED CODE.

And, since :RELEASEing a file gives everyone write and execute access to it,

* *NEVER* :RELEASE A PROGRAM FILE THAT RESIDES IN A GROUP WHICH HAS PM CAPABILITY!

As if this wasn't enough, there are some other potential security violations that can occur with privileged mode. Consider the following circumstance:

Two HP 3000s, which we will call O-machine (intended for OPEN access) and S-machine (which the system management wants SECURED) are linked via DS/3000. A person has a userid and a group with PM capability on O-machine and a plain vanilla userid and group with only default capabilities on S-machine. S-machine management thinks that their machine is secure, since only MANAGER.SYS and PUB.SYS have PM capability on their machine.

Now, there are several file system operations that bypass system security and thus require privileged mode; for instance:

* FOPEN with the 4 low-order bits of aoptions set to 15 (see ASIDE -- ALLOWING PROGRAMS TO READ :SECURE'D FILES), when called from within privileged mode, lets you read a file even when you have no access to it.

* FOPEN with EXECUTE access (4 low-order bits of aoptions set to 6; document in System Intrinsics manual), when called from within privileged mode, lets you read and write a file if you have only execute access to it.

* MUSTOPEN, a procedure identical to FOPEN in all respects except that, when called in privileged mode, it ignores a file's lockword.

* FOPEN of a privileged file (a file with a negative filecode, such as an IMAGE database).

These are not inherently security violations -- in fact, as the ASIDE -- ALLOWING PROGRAMS TO READ :SECURE'D FILES section shows, they can be used to actually INCREASE your security. However, they are not security violations only because they require PM capability to be executed.

Now, consider our would-be security violator. He has his eyes on the S-system file FOO.JOB.SYS, which he knows is a job stream that contains an embedded password (it could just as well contain any other kind of sensitive data). He signs on to O-system as a privileged user, and then to

the S-system via DS as a plain vanilla user. Now, because, DS allows a program on one system to open a file on another system (by specifying the file's device to be the dsline device followed by a "#", e.g. "60#"), our user writes a program on O-system that opens file FOO.JOB.SYS in the "ignore security mode" (aoptions 4 low-order bits = 15) on S-system. Since the program is running in privileged mode (remember, our O-system user is privileged), the open succeeds, and the user can read the file!

Now note that the file system does not check that the user on S-system must have PM capability to use this security-bypassing mode; the program need merely be running in PM capability, regardless of which system it is on!

This is one of the few genuine flaws in MPE's security system, and it's nothing to sneeze at. What it means is that

* IF TWO HP3000'S ARE CONNECTED VIA DS, AND A USER HAS PM CAPABILITY ON ONE AND AN ORDINARY LOGON ON THE OTHER, HE CAN VIOLATE THE OTHER'S SECURITY. THUS, IF ANY HP3000 IN A DS NETWORK IS BROKEN INTO OR LEFT OPEN, ALL OTHERS ARE IN GRAVE DANGER.

Thus, if you want to keep one system secure, you must keep all systems hooked up to it via DS secure as well.

One other issue, somewhat more arcane but nonetheless relevant arises when using privileged mode.

If a program which has PM capability calls DEBUG when the user running it does not have PM capability, even though the user will be dropped into non-privileged DEBUG, he can use it to break system security.

Briefly, the user can modify some data in the program's stack or the program's P pointer (which points to the current instruction being executed) to cause the program to do something other than what is supposed to do when it performs its privileged operations. One thing that actually happened to one of my programs is that it called the WHO intrinsic, figured out the logon user, account, and group, put them into global arrays, and then went into privileged mode and got the logon user, account, and group passwords and wrote them to a`stream file. This was perfectly kosher -- if a user managed to sign on, he already knows his logon passwords; however, the program allowed the user to enter DEBUG even if he was non-privileged. Although the program did not call DEBUG when privileged, and the user was not put into privileged debug, the user could modify the user, account, and group id arrays in the stack to read, say, "MANAGER", "SYS", and "PUB". Then, the next stream the program built would contain MANAGER.SYS's passwords!

This is, as I said, a rather arcane and relatively infrequent problem; however, it is a possible security flaw nonetheless, and should not be ignored. In fact, I'd like to ask HP to correct their DBDRIVER program, which is privileged and has a "/D" command which drops the user into DEBUG whether or not he is privileged.

In the same vein, dynamically loading (via the LOADPROC intrinsic) a procedure from a user's group or account SL and then calling it should also be forbidden to privileged programs -- the called procedure, even though it resides in a non-privileged SL, can call GETPRIVMODE because the program calling it is privileged. Again, rather arcane but still worth noting.

Thus,

* PRIVILEGED PROGRAMS MUST NEVER CALL DEBUG UNLESS THEIR USER IS PRIVILEGED, AND MUST NEVER DYNAMICALLY LOAD AND CALL PROCEDURES FROM A USER'S GROUP OR ACCOUNT SL UNLESS THE USER IS PRIVILEGED.

Now, I do not intend to unfairly malign PM capability. It has its uses, and in fact, some programs must have it (such as the HP system utilities in PUB.SYS or many very useful contributed and vendor-supported programs). However, and I can not stress this enough, use of PM must be watched very carefully if you wish to keep your system secure.

IGNORANCE SECURITY

Many techniques of violating system security described herein may appear rather complicated and improbable; in fact, they are. It is all too easy to say: "Well, my users aren't so smart -- they'd never think of pulling all those tricks". Unfortunately, it is of such complacency that insecure systems are born. After all, if we could think of these tricks, why can't some smart guy in your shop? What if he reads this paper? What if one of his friends is a sophisticated HP user? The assets of your

company are far too precious a thing to entrust to the presumed ignorance of your users; you should rather improve the security of your system, so that even a smart user will not be able to penetrate it -- and if your users aren't that smart, all the better.

DATABASE SECURITY

IMAGE/3000's security system is probably one of its most complex features and also one of its least used. My first impulse was to chastise the HP user community for not using this wonderful security feature more, and to blame 99.44% of all security violations on their failure to do so, but then I realized that this is not such a wonderful facility after all.

IMAGE/3000 security permits the database creator to restrict access to each individual data item and data set to only those users who specify a certain password when opening the database. Admittedly, this is a very useful feature when you expect the database to be accessed via QUERY -- then you can define exactly what a user can do by what password you give him. However, most databases are accessed by application programs, not through QUERY, and most of the time it is the program, not the user, that specifies the password. So, unless you intend to reveal certain database passwords to only certain programmers and thus protect your database against your programmers, not your users, you are probably far better off implementing application security, i.e. having your application figure out what a certain user is authorized or not authorized to do, rather than using IMAGE security.

* IMAGE/3000 DATABASE SECURITY IS NOT PARTICULARLY USEFUL EXCEPT FOR PROTECTING DATABASES AGAINST UNAUTHORIZED QUERY ACCESS. IN FACT, SOME DEGREE OF PROTECTION AGAINST UNAUTHORIZED QUERY ACCESS CAN BE GIVEN BY USING DBUTIL'S "SET SUBSYSTEM" COMMAND TO DISALLOW ANY QUERY ACCESS OR QUERY MODIFICATION OF A DATABASE.

DATA ENCRYPTION

If you want to secure your data against unauthorized reading, you need not prevent anybody from accessing it if, even if they manage to access it, they won't be able to understand it. This is the principle of encryption -- change the format of your data so that nobody except for the authorized people will be able to understand it.

Usually, encryption algorithms involve the use of so-called "keys". Say that I want to encrypt the phrase "NOW IS THE TIME FOR ALL GOOD MEN TO COME TO THE AID OF THEIR COUNTRY". I could do this by choosing some number (say, 7) and adding it to each letter of the sentence, so that A would become H, B would become I, C would become J, R (#18) would become Z, S would become A, etc. Then, the phrase would become "UVD PZ AOL APTL MVY HSS NVVK TLU AV JVTL AV AOL HPK VM AOLPY JBVUAYF", an unreadable jumble of letters to anyone who doesn't know that to decrypt it, one must subtract 7 from each character. Thus, 7 is the key and the encryption algorithm is to add the key to each character.

Unfortunately, things are a bit more complicated than that, primarily because with some work, one can realize that the letters A and V occur quite often, the combination AO occurs frequently as well, and that there are only so many possible two-letter words (some of which must correspond to PZ, AV, and VM). Thus, we could find out what key letters correspond to, and thus decode the entire sentence.

Fortunately, there are more sophisticated encryption algorithms that are far harder to decrypt. And, since the key need not be stored on the computer, but only in the user's mind or some other safe place, encrypted data can only be decrypted by an authorized person.

Another, less general but nonetheless useful technique for encrypting passwords is called "one-way encryption". Say that you wish a user to enter a password into your program when he is first set up, and then have your program ask him for the password every time he subsequently logs on. You do not need to actually decrypt the password -- just encrypt it once at user set-up time, store it in encrypted form, and then, every time the user tries to log on, ask him for a password, encrypt his answer, and compare it against the encrypted real password.

Thus, your encryption algorithm can map the entire password into a single number (by, say, adding the squares of all the letters, each multiplied by the cube of its position in the password string), thus making it impossible to decrypt; and, the encryption algorithm is much simpler than two-way encryption algorithms that need to have a corresponding decryption

algorithm. Unfortunately, this technique is limited to applications in which decryption is never necessary, such as when passwords are stored.

SECURITY

One-way encryption is easy to do; good two-way encryption is harder -- I know of no HP programs that do it, but hopefully that will be remedied soon.

* IN GENERAL, ENCRYPTION IS ANOTHER GOOD WAY OF PROTECTING SENSITIVE DATA FROM UNAUTHORIZED READING.

CONCLUSION

It is all too easy to get involved in the implementation and perfection of an application system, putting "little things" like security on the back burner; unfortunately, this is precisely what accounts for the alarming amount of computer crime that is threatening us today. What is best is that with application of some simple guidelines and a little time and effort, you could dramatically decrease your chances of becoming a victim. No security system will cut these chances to zero, but if you have as much valuable data in your machine as the average HP user has in his, doing nothing can literally cost you millions.

APPENDIX A: SUMMARY OF USEFUL HINTS

* THE USER IS THE WEAKEST LINK IN THE LOGON SECURITY SYSTEM -- DISCOURAGE HIM FROM REVEALING PASSWORDS (by techniques such as personal profile security or even by reprimanding people who reveal passwords -- they seem innocent, but they can lose you millions).

* PASSWORDS EMBEDDED IN JOB STREAMS ARE EASY TO SEE AND VIRTUALLY IMPOSSIBLE TO CHANGE -- AVOID THEM.

* SOME FORMS OF ACCESS ARE INHERENTLY SUSPECT (AND THUS REQUIRE EXTRA PASSWORDS) OR ARE INHERENTLY SECURITY VIOLATIONS. THUS, ACCESS TO CERTAIN USER ID'S AT CERTAIN TIMES OF DAY, ON CERTAIN DAYS OF WEEK, AND/OR FROM CERTAIN TERMINALS (SUCH AS DIAL-IN OR DS LINES) SHOULD BE SPECIALLY RESTRICTED.

* MANY SECURITY VIOLATIONS CAN BE AVERTED BY MONITORING THE WARNINGS OF UNSUCCESSFUL VIOLATION ATTEMPTS THAT OFTEN PRECEDE A SUCCESSFUL ATTEMPT. IF POSSIBLE, CHANGE THE USUAL MPE CONSOLE MESSAGES SO THEY WILL BE MORE VISIBLE.

* LEAVING A TERMINAL LOGGED ON AND UNATTENDED IS JUST AS MUCH A SECURITY VIOLATION AS REVEALING THE LOGON PASSWORD. USE SOME KIND OF TIMEOUT FACILITY TO ENSURE THAT TERMINALS DON'T REMAIN INACTIVE FOR LONG; SET UP ALL YOUR DIAL-IN TERMINALS WITH SUBTYPE 1.

* A USEFUL APPROACH TO SECURING YOUR SYSTEM IS TO SET UP A LOGON MENU WHICH ALLOWS THE USER TO CHOOSE ONE OF SEVERAL OPTIONS RATHER THAN TO LET THE USER ACCESS MPE AND ALL ITS POWER DIRECTLY.

* BLOCKING OUT MPE COMMANDS VIA UDC'S WITH THE SAME NAME WILL USUALLY FAIL UNLESS THE COMMAND IS :SETCATALOG OR :SHOWCATALOG OR IF YOU ALSO FORBID ACCESS TO MANY HP SUBSYSTEMS AND HP-SUPPLIED PROGRAMS. THIS SEVERELY LIMITS THE USEFULNESS OF THIS METHOD.

* REMEMBER THAT :RELEASE'ING A FILE LEAVES IT WIDE OPEN FOR ANY KIND OF ACCESS; :RELEASE FILES CAUTIOUSLY, AND RE-:SECURE THEM AS SOON AS POSSIBLE.

* TRY TO MAKE IT AS EASY AS POSSIBLE FOR PEOPLE TO ALLOW THEIR FILES TO BE ACCESSED BY OTHERS WITHOUT HAVING TO :RELEASE THEM. THUS, BUILD

ALL ACCOUNTS WITH (R,W,X,A,L:ANY) SO THAT ALLOWING ACCESS TO A GROUP WILL BE EASIER.

* IF A GROUP IS MOSTLY COMPOSED OF FILES THAT SHOULD BE ACCESSIBLE BY ALL USERS IN THE SYSTEM OR BY ALL ACCOUNT USERS, BUILD IT THAT WAY. THIS WILL ALSO REDUCE :RELEASE'S.

* THE :ALTSEC COMMAND IS USEFUL FOR RESTRICTING ACCESS TO FILES IN A GROUP TO WHICH ACCESS IS NORMALLY LESS RESTRICTED.

* LOCKWORDS AREN'T ALL THEY'RE CRACKED UP TO BE. OTHER APPROACHES SHOULD BE PREFERRED.

APPENDIX A: SUMMARY OF USEFUL HINTS

* YOU SHOULD ONLY GIVE OP CAPABILITY TO USERS WHO YOU TRUST AS MUCH AS YOU WOULD A SYSTEM MANAGER, TO USERS WHO HAVE NO ACCESS TO MAGNETIC TAPES OR SERIAL DISCS, OR TO USERS WHO HAVE A LOGON UDC THAT DROPS THEM INTO A MENU WHICH FORBIDS THEM FROM DOING :STORE'S OR :RESTORE'S

* YOU SHOULD ONLY GIVE PM CAPABILITY TO USERS WHO YOU TRUST AS MUCH AS YOU WOULD A SYSTEM MANAGER.

* IF ANY USER HAS SAVE ACCESS TO A GROUP WITH PM CAPABILITY, OR WRITE AND EXECUTE ACCESS TO ANY PROGRAM FILE THAT RESIDES IN A GROUP WITH PM CAPABILITY, HE CAN WRITE AND RUN PRIVILEGED CODE.

* *NEVER* :RELEASE A PROGRAM FILE THAT RESIDES IN A GROUP WHICH HAS PM CAPABILITY!

* IF TWO HP3000'S ARE CONNECTED VIA DS, AND A USER HAS PM CAPABILITY ON ONE AND AN ORDINARY LOGON ON THE OTHER, HE CAN VIOLATE THE OTHER'S SECURITY. THUS, IF ANY HP3000 IN A DS NETWORK IS BROKEN INTO OR LEFT OPEN, ALL OTHERS ARE IN GRAVE DANGER.

* PRIVILEGED PROGRAMS MUST NEVER CALL DEBUG UNLESS THEIR USER IS PRIVILEGED, AND MUST NEVER DYNAMICALLY LOAD AND CALL PROCEDURES FROM A USER'S GROUP OR ACCOUNT SL UNLESS THE USER IS PRIVILEGED.

* IMAGE/3000 DATABASE SECURITY IS NOT PARTICULARLY USEFUL EXCEPT FOR PROTECTING DATABASES AGAINST UNAUTHORIZED QUERY ACCESS. IN FACT, SOME DEGREE OF PROTECTION AGAINST UNAUTHORIZED QUERY ACCESS CAN BE GIVEN BY USING DBUTIL'S "SET SUBSYSTEM" COMMAND TO DISALLOW ANY QUERY ACCESS OR QUERY MODIFICATION OF A DATABASE.

* IN GENERAL, ENCRYPTION IS ANOTHER GOOD WAY OF PROTECTING SENSITIVE DATA FROM UNAUTHORIZED READING.

BIOGRAPHY

Eugene Volokh was born February 29, 1968 in Kiev, USSR. His family moved to the U.S.A. in 1975. In 1979 Eugene started his work with HP equipment. Eugene is now a senior consultant at VESOFT, Inc., a company he cofounded with his father, Vladimir Volokh, in 1980 (their products are MPEX/3000 and SECURITY/3000). This year Eugene graduated from UCLA with a B.S. degree in computer science.

OPT/3000 - What It Is & What It Does

by Tom Idema

I. Introduction

A. What is OPT/3000?

OPT/3000 is Hewlett-Packard's On-line Performance Tool for use with HP3000 computers using the MPE operating system. Its primary uses include performance measurement, system utilization and tuning. As as package, OPT/3000 consists of two interdependent products, the OPT/3000 Software and OPT/3000 System Performance Training Course. It is designed to be interactive, although it has certain batch capabilities and on-line help capabilities as well. OPT/3000 is a powerful tool to aid the trained user in monitoring and improving the performance of the HP3000.

B. How does OPT/3000 help me?

OPT/3000 is an invaluable tool which allows the system manager to look deep inside the HP3000 and monitor almost everything going on. OPT/3000 is used in our facility for collecting system utilization data and for characterizing system performance by charting and comparing current data with previous samples over time. I've found OPT/3000 to be far superior to the old "crystal ball" approach, (which was about all that was available to system managers prior to the introduction of OPT/3000 as a product), especially when it comes to isolating problem areas, be it disc I/O, memory, or whatever.

In the area of system management, OPT/3000 is used to monitor the system performance and assist with system fine tuning. It helps with the identification of processing bottle necks, helps improve overall system preformance and helps in the area of capacity planning. For instance, prior to the introduction of MPE IV, our HP3000 Series III was showing signs of saturation with an average CPU utilization of around 50% during a 24 hour work day as shown in Figure 1; and, indications were that a more powerful CPU would be needed in the near future. After installing MPE IV on this (*) machine, however, the performance data obtained from OPT/3000 indicated a CPU utilization of slightly over 25%, which meant that the replacement of the current operating equipment could be postponed for some time (See Figure 2). (*)

Although this is only one example, it should be noted that with OPT/3000 almost all aspects of the HP3000 can be monitored and it allows you to consider your system as a whole.

II. OPT/3000 Functions

OPT/3000 can generate over twenty unique displays, each showing a different aspect of system performance data. These are grouped into six major categories or functions which will be discussed briefly.

A. Globals

The Globals function shows summary level information describing present CPU usage, memory utilization, disc I/O rates (Figure 3),

and a summary of jobs and sessions currently running (Figure 4). (*)

These are two Global displays which allow you to quickly identify potential problem areas or to monitor general system activity and determine trends in resource usage. For instance, current CPU utilization can be found by adding CPU Busy and the Overhead percentages. Figure 5 shows a HP3000 series 44 that is 86% busy with 14% of Overhead...100% current CPU utilization. (*)

However, there was a problem program running at this time which was in a hard loop. Figure 6 shows the same CPU only 16% busy seven minutes later after the problem program was aborted. (*)

A hard copy summary report is also available within the Global function which provides an overall view of the system (See Figure 7). (*)

This can be generated interactively or in batch mode. It is from 1/2 hour summary reports generated on a random 24 hour basis that I plot the average CPU utilizations such as shown in Figure 8; in this case an HP3000 series 64 with 4MB of memory and over 35 interactive terminals. (*)

After consulting the Global displays, which are presented first when executing OPT/3000, more detailed displays from the other functional areas can be used for isolation or verification of potential performance problems.

B. Memory

The Memory function has displays which provide information about the usage of memory and its segmentation. These displays provide not only the use and contents of memory, but histograms as to the size and distribution of code segments, stack data, etc. The entire contents of memory can be displayed, or that of a specific bank, according to your needs.

Figure 9 shows the summary usages of all memory, linked memory and code, stack and data segments. Should a high percentage of linked memory be locked or frozen, then the memory manager could have problems finding space, depending on the location of the frozen area. (*)

Memory contents shown in Figure 10 provide a clearer picture of overall memory content and usage. Note the frozen areas in Bank 01 indicated by the "///'s". In this case they present no problem due to location. (*)

Should a detailed image of a specific bank of memory be required, Figure 11 provides an example. (*)

Figure 12 displays three of the histograms available for analysis of code, stack and extra data segments. If the charts indicate a large number of segments over 10K in size, the memory manager may have problems in satisifying requirements for absent segments. (*)

C. CPU-Memory Manager

This function provides information related to CPU usage and memory management activity as percentages of time in various states and process execution rates. The various rates include CPU time for execution, memory management, overhead processing, waiting and CPU idle time.

Notice in Figure 13, the CPU Usage Display, that "paused for swap" is not present, indicating that memory is not a problem at this time, but that the "paused for disc" is greater than the 10% optimum range indicating that disc I/O may, indeed, be a problem and bears watching. (*)

The other displays, Figures 14 & 15, support the fact that memory is probably sufficient for the current workload on the system. (*)

D. I/O Function

The I/O displays provide disc I/O completion rates and data relative to printer and tape activity as well. The I/O completion rate for each type of device is displayed for both the current and overall time intervals. Information can be displayed down to a specific device and allows you to determine the balance of your I/O load across devices.

The I/O Activity Report provides an overall I/O summary by device type as shown in Figure 16. In this case only user disc I/O, at the rate of 54 per second, is taking place. (*)

A closer look at disc drives, 1 through 4 in Figure 17, shows both the overall performance and the Read/Write activity currently taking place on the listed devices. Here it is important to look at the distribution of the I/O load to see if certain devices are getting all of the activity, and then to see what applications or files are in use to determine if better file placement might decrease disc contention, etc. (*)

E. Processes

The Process function provides information about process and program activity on the system, including file names, file sizes, program segments, number of users and working set data. Detailed information regarding each process is also available including process stack and space utilization.

The Program File Display in Figure 18 identifies all program files which are allocated or currently in use. The "#PS" in this case shows the number of process sharing a given program file. Figure 19, the User Summary Display gives the detail with (*) regard to these same processes. Note PIN #115; this is the process with the 28K stack as shown in figure 12 displayed earlier. (*)

The Process State Report summarizes the information about all processes known to the system as shown in Figure 20. Here, if the dispatch wait lists are greater than 5, it is possible to have CPU contention; disc contention if the short wait list is greater than 5. (*)

The Process Display, Figure 21, is the most detailed of the process function displays. In this display stack utilization, system status, files, data segments and even stack marker information can be analyzed. (*)

F. System Tables

The System Tables function has two displays which provide both the current and maximum utilization of all configurable system tables.

These displays give you an opportunity to see what the present and past table utilization has been, and from that establish an optimum set for your system which minimizes both memory usage and the risk of system failures caused by tables configured too small.

Tables, (See Figure 22), which consistantly have low utilization of thirty percent or less over a lengthly period of time, could possibly be reconfigured smaller, and thereby free up real memory. However, peak loads must be considered before reduction of table sizes. Conversely, tables with consistantly high rates of utilization (over ninety percent), could cause poor performance and probably should be made larger in order to improve system performance. (*)

Figure 23, shows a graphic representation of the table usages and provides a quick and dirty indication of present and peak usages shown in detail in Figure 22. (*)

III. System Performance Training Course

A. An HP Requirement

Hewlett Packard requires that at least one person from a site attend their eight day course at initial installation of the OPT/3000 package.

B. The Course Covers

The eight day System Performance Training Course required by Hewlett Packard is conducted by an HP Performance Specialist, and covers the internals of the MPE operating system and the techniques of performance analysis. Proper interpretation of OPT/3000 data requires an understanding of the MPE operating system that can only be obtained through such a course.

The internals course is required because OPT/3000 presents detailed information which often must be analyzed or interpreted by the System Manager. The operating system and processes, their tables, relationships and uses are covered in great detail along with the memory manager, scheduler, dispatcher and system I/O.

Time is also spent on the functions of the file system and in performance measurement where system configurations, scheduling and operation management are covered, along with tuning, upgrade planning, software evaluation and general control of system resources.

IV. Conclusion:

A. Cost

1. OPT/3000 software 32238A.....$6,400

2. System Performance Training 22809B.....$1,640

B. Is It For You ?

HOW'S YOUR SYSTEM PERFORMANCE ??

Personally, I'd be lost without OPT/3000. I maintain all three of my HP3000's on a regular basis, and as a result, have been able to keep them quite well tuned; and my performance has been good. But, OPT/3000 has also helped me to isolate and/or avoid various problems before they have become serious and caused the system to degrade.

OPT/3000 is just one tool and by itself will not substitute for the regular and deliberate application of "system management" functions such as disc management, scheduling, housekeeping, and good application design. But, with OPT/3000, the System Manager has one very powerful tool with which to "manage and monitor" the HP3000 computer systems.

(*) Pertains to diagrams for the document.

Biographical Information

of

Tom Idema

Tom Idema is Manager of MIS Technology Services for the Furniture Systems Division of Westinghouse Electric Corporation. He is a graduate of Ole Miss, with an MBA from Western Michigan University. Tom served with the Marines and flew jets in Viet Nam before launching his data processing career first with General Foods, and later with Hewlett-Packard, prior to joining Westinghouse. He has served as President of both the Westinghouse Corporate HP Users' Group and the Lake Michigan Regional Users' Group which serves the western Michigan area.

Tom is a member of the faculty of Grand Rapids Junior College, and has taught data processing classes for the past nine years. He has also had several articles published in a national data processing journal.

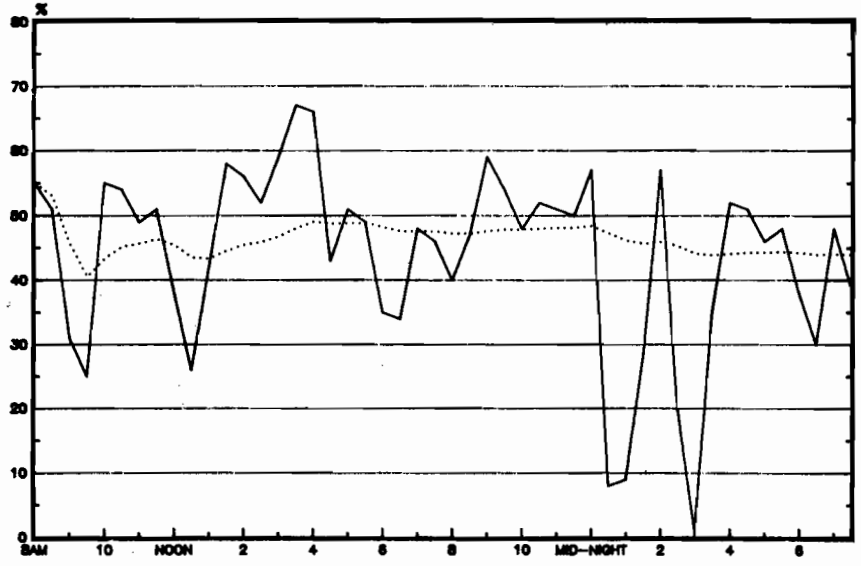
For the past two years he has served as a member of the HPIUG Affiliate Council.

M.I.S. TECHNICAL SERVICES

SYSTEM "A" CPU UTILIZATION

PERCENT BUSY

AVG. PCT. BUSY



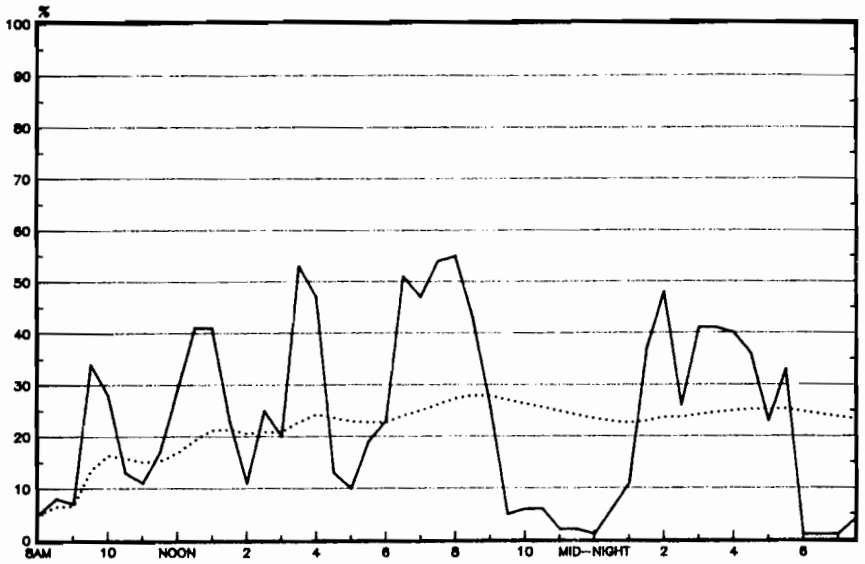
MAY 18, 1982

M.I.S. TECHNICAL SERVICES

SYSTEM "A" CPU UTILIZATION

PERCENT BUSY

AVG. PCT. BUSY



NOVEMBER 10, 1982

```
    PROCESS SUMMARY DISPLAY      HP32238A.00.10  OPT/3000                    MON, OCT 31, 1983, 11:29 AM       PAGE   2
(C) HEWLETT-PACKARD COMPANY 1979, 1980                  WESTINGHOUSE SYSTEM 'C'
NUMBER OF CURRENT PROCESSES: 90

                    10        20        30        40        50        60        70        80        90        100
                   ----------------------
  ACTIVE SESSIONS  !                    !                                                                    (22/22)
                   ----------------------
                   ----
      ACTIVE JOBS  !  !                                                                                      (3/3)
                   ----


                   ----------------------
 SYSTEM PROCESSES  !                    !                                                                    (21/21)
                   ----------------------
                   -----
    CI PROCESSES   !   !                                                                                     (4/4)
                   -----
                   ----------------------
   USER PROCESSES  !                    !                                                                    (21/21)
                   ----------------------
                   -----------------------
CREATED PROCESSES  !                     !                                                                  (23/23)
                   -----------------------


                   -----
IN DISPATCH QUEUE  !   !                                                                                     (4/4)
                   -----
                   ------------------------------------------------------------------------------------------
      IN NO QUEUE  !                                                                                        !  (86/86)
                   ------------------------------------------------------------------------------------------




LEGEND:    ---
           ! ! CURRENT COUNT
           ---

           ---
           !/! MAXIMUM OBSERVED WHILE IN THIS DISPLAY
           ---
```

```
     RESOURCE USAGE DISPLAY      HP32238A 00.10  OPT/3000                    MON, OCT 31, 1983, 11:29 AM       PAGE   1
(C) HEWLETT-PACKARD COMPANY 1979, 1980                    WESTINGHOUSE SYSTEM 'C'
   CURRENT INTERVAL     11.7 SECONDS          OVERALL INTERVAL    105.6 MINUTES

ACTIVITY IN CURRENT INTERVAL

                          10        20        30        40        50        60        70        80        90       100%
      MEMORY USAGE MMC----------------------------------CS--------------------------SD----------D          (2/40/32/18)

         CPU STATE B-------------------------------------------------BP-----------------------------PO----------O (50/36/14)

                          10        20        30        40        50        60        70        80        90       100 per second
 DISC I/O ACTIVITY U-------------------------------------------U                                                  (54)
-------------------------------------------------------------------------------------------------------------------------------
ACTIVITY OVER ALL INTERVALS
                          10        20        30        40        50        60        70        80        90       100%
         CPU STATE B----------------------------------BP--------------------------------PI------IO----------O (42/38/8/12)

                          10        20        30        40        50        60        70        80        90       100 per second
 DISC I/O ACTIVITY U-------------------------------------------U                                                  (50)



MEMORY USAGE LEGEND:
  M  Resident MPE
  C  Code segments
  S  Stack segments
  D  Data segments

CPU STATE LEGEND:

  B  Busy on processes
  P  Paused for user and/or memory management disc I/O
  I  Idle
  G  Garbage collection
  O  Memory allocation and ICS overhead

DISC I/O ACTIVITY LEGEND:

  U  User disc I/O
  M  Memory management disc I/O
```

```
      RESOURCE USAGE DISPLAY      HP32238A.00.10  OPT/3000                    FRI, SEP  9, 1983,  5:57 PM       PAGE   1
(C) HEWLETT-PACKARD COMPANY 1979, 1980                    SHPHUNG
   CURRENT INTERVAL:     15.9 SECONDS          OVERALL INTERVAL:      5.3 MINUTES

ACTIVITY IN CURRENT INTERVAL

                          10        20        30        40        50        60        70        80        90        100%
      MEMORY USAGE MMC----------------------------------CS--------------------SD----------D                    (2/36/24/14

         CPU STATE B-------------------------------------------------------------------------------------BO----------O (86/14)

                          10        20        30        40        50        60        70        80        90        100 per second
 DISC I/O ACTIVITY UU                                                                                              (2)
-------------------------------------------------------------------------------------------------------------------------------
ACTIVITY OVER ALL INTERVALS
                          10        20        30        40        50        60        70        80        90        100%
         CPU STATE B-------------------------------------------------------------------------------------------BO--O (96/4)

                          10        20        30        40        50        60        70        80        90        100 per second
 DISC I/O ACTIVITY U--U                                                                                            (4)



MEMORY USAGE LEGEND:
  M  Resident MPE
  C  Code segments
  S  Stack segments
  D  Data segments

CPU STATE LEGEND:

  B  Busy on processes
  P  Paused for user and/or memory management disc I/O
  I  Idle
  G  Garbage collection
  O  Memory allocation and ICS overhead

DISC I/O ACTIVITY LEGEND:

  U  User disc I/O
  M  Memory management disc I/O
```

```
      RESOURCE USAGE DISPLAY      HP32238A.00.10  OPT/3000                        FRI, SEP  9, 1983,  6:04 PM       PAGE   4
(C) HEWLETT-PACKARD COMPANY 1979, 1980                      SHPHUNG
   CURRENT INTERVAL:      3.8 SECONDS          OVERALL INTERVAL:     12.1 MINUTES

ACTIVITY IN CURRENT INTERVAL

                          10        20        30        40        50        60        70        80        90        100%
      MEMORY USAGE MMC----------------------------CS------------------SD----------D                                (2/34/22/14)

         CPU STATE B--------------BI-------------------------------------------------------------------------------I (16/88)

                          10        20        30        40        50        60        70        80        90        100 per second
 DISC I/O ACTIVITY UU                                                                                               (2)
-------------------------------------------------------------------------------------------------------------------------------
ACTIVITY OVER ALL INTERVALS
                          10        20        30        40        50        60        70        80        90        100%
         CPU STATE B-------------------------------------------------------------------------------BPPIIO----O (80/2/2/6)

                          10        20        30        40        50        60        70        80        90        100 per second
 DISC I/O ACTIVITY U------------U                                                                                   (14)



MEMORY USAGE LEGEND:
  M  Resident MPE
  C  Code segments
  S  Stack segments
  D  Data segments

CPU STATE LEGEND:

  B  Busy on processes
  P  Paused for user and/or memory management disc I/O
  I  Idle
  G  Garbage collection
  O  Memory allocation and ICS overhead

DISC I/O ACTIVITY LEGEND:

  U  User disc I/O
  M  Memory management disc I/O
```

SUMMARY REPORT HP32238A.00.10 OPT/3000 MON, OCT 31, 1983, 11:48 AM REPORT 9
(C) HEWLETT-PACKARD COMPANY 1979, 1980 WESTINGHOUSE SYSTEM 'C'
INTERVAL LENGTH: 1045.867 SECONDS (17.4 MINUTES)

CPU ACTIVITY SUMMARY

CPU STATE	MEAN	MAX	LENGTH	COUNT	TOTAL TIME
CPU BUSY	59%	71%	.006	107426	605.737
PAUSE DISC & SWAP	1%	5%	.023	314	7.219
PAUSE DISC	17%	54%	.019	9397	181.895
PAUSE SWAP	0%	0%	.020	8	.163
PAUSE IDLE	2%	30%	.084	404	25.771
GARBAGE COLLECTION	0%	0%		0	.000
MEMORY ALLOCATION	0%	1%	.003	444	1.473
ICS OVERHEAD	21%				223.609

MEMORY ALLOCATION SUMMARY

RESULT	MEAN	COUNT
RECOVERY	1%	2
FREE SPACE	99%	372
OVERLAY CAND	0%	0
GIVE UP	0%	0
HARD REQUEST	0%	0

LAUNCH ACTIVITY AND ADDITIONAL MEMORY MANAGEMENT ACTIVITY SUMMARY

	PROCESS LAUNCHES	PROCESS SWAP-INS	PROCESS PREEMPTS	MEMORY ALLOCS	SPECIAL REQUESTS	MM I/O READS	MM I/O WRITES	RELEASE DATA SEG	RELEASE CODE SEG	CLOCK CYCLES
COUNT	107426	444	33963	373	175	371	115	0	0	0
RATE	102.7	.4	32.5	.4	.2	.4	.1	.0	.0	.0
MAX RATE	128	2	46	2	1	2	1	0	0	0

SUMMARY OF DISC ACTIVITY

	COUNT/RATE ALL I/O	READS	WRITES	CONTROL OPS	MAXIMUM RATE (USER/MM) READS	WRITES	CONTROL OP
ALL DISC	24488/ 23.4	17247/ 16.5	6766/ 6.5	475/ .5	33/ 2	12/ 1	2
DISC 1 (LDEV 1)	3427/ 3.3	3027/ 2.9	366/ .3	34/ .0	19/ 0	2/ 0	1
DISC 2 (LDEV 2)	2567/ 2.5	1892/ 1.8	642/ .6	33/ .0	5/ 0	4/ 0	0
DISC 3 (LDEV 3)	2998/ 2.9	1796/ 1.7	1155/ 1.1	47/ .0	14/ 1	3/ 1	0
DISC 4 (LDEV 4)	7550/ 7.2	6391/ 6.1	1068/ 1.0	91/ .1	22/ 1	2/ 0	1
DISC 5 (LDEV 5)	2001/ 1.9	877/ .8	1072/ 1.0	52/ .0	2/ 0	2/ 0	0
DISC 6 (LDEV 11)	1408/ 1.3	727/ .7	608/ .6	71/ .1	4/ 0	2/ 0	1
DISC 7 (LDEV 12)	2805/ 2.7	1535/ 1.5	1188/ 1.1	82/ .1	17/ 0	3/ 0	0
DISC 8 (LDEV 13)	1734/ 1.7	1002/ 1.0	667/ .6	65/ .1	2/ 0	1/ 0	0

SUMMARY OF LP ACTIVITY

	COUNT/RATE ALL I/O	WRITES	CONTROL OPS	MAXIMUM RATE WRITES	CONTROL OP
ALL LP	0/ .0	0/ .0	0/ .0	0	0

SUMMARY OF TAPE ACTIVITY

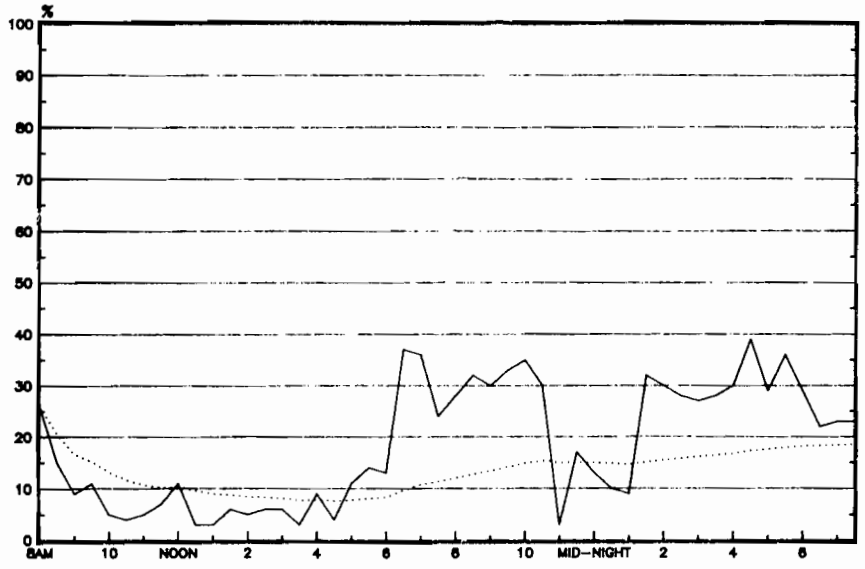
	COUNT/RATE ALL I/O	READS	WRITES	CONTROL OPS	MAXIMUM RATE READS	WRITES	CONTROL OP
ALL TAPE	61/ .1	0/ .0	61/ .1	0/ .0	0	0	0
TAPE 1 (LDEV 7)	61/ .1	0/ .0	61/ .1	0/ .0	0	0	0

M.I.S. TECHNOLOGY SERVICES

SYSTEM "A" CPU UTILIZATION

PERCENT BUSY

AVG. PCT. BUSY



AUGUST 8, 1983

```
      MEMORY REPORT DISPLAY     HP32238A.00.10  OPT/3000     WESTINGHOUSE SYSTEM 'C'     MON, OCT 31, 1983, 11:36 AM     PAGE  14
(C) HEWLETT-PACKARD COMPANY 1979, 1980
32 BANKS     LINKED MEMORY: 2035584WORDS

          ALL MEMORY M-MC----10--------20--------30--------40--------50--------60--------70--------80--------90--------D    100%   (3/38/31/20)
        LINKED MEMORY F-FA----------------------------------------------------------------------------------------A           (3/89)
        CODE SEGMENTS U-------------------------------US-------------------------------------------------------S (55/45)
                      P-----------------------PC-----------------------------------------------------------C (35/65)
       STACK SEGMENTS U-------------------------------------------------------------------------------------US--S (90/4)
        DATA SEGMENTS U-------------------------------------------------------------------------------------US--S (95/5)

LEGEND:
ALL MEMORY - Includes all of memory

    M  Resident MPE
    C  Code Segments
    S  Stack Segments
    D  Extra Data Segments

LINKED MEMORY - All memory except for Resident MPE
    A  Active Segments
    O  Segments marked as Overlay Candidates
    F  Frozen, Locked, or I/O Frozen Segments

CODE, STACK, & EXTRA DATA SEGMENTS

    U  User Segments
    S  System Segments
    P  Program File Segments
    C  Code Segments from CST
    M  Non-special Extra Data Segments  (usage unknown)
    F  File System Segments
    J  Job Management Segments
    I  Image Segments
    K  KSAM Segments
    T  System Table Segments  (only those that are in linked memory)
```

MEMORY CONTENTS DISPLAY HP32238A 00.10 OPT/300D WESTINGHOUSE SYSTEM 'C' MON, OCT 31, 1983, 11:35 AM PAGE 12
(C) HEWLETT-PACKARD COMPANY 1978, 1980
32 BANKS LINKED MEMORY 2035584WORDS
Resident MPE

```
     MEMORY CONTENTS DISPLAY      HP32238A.00.10  OPT/3000                                MON, OCT 31, 1983, 11:35 AM       PAGE  13
(C) HEWLETT-PACKARD COMPANY 1979, 1980                        WESTINGHOUSE SYSTEM 'C'

LEGEND AND SUMMARY STATISTICS

Each space represents 1K words of memory (rounded to the nearest 1K words, with segments smaller than 1K always rounded up to 1K)
Lower case denotes a system segment
          SEGMENT  % OF LINKED
SYMBOL     COUNT      MEMORY     DESCRIPTION
-------------------------------------------------------------------------------
   S          48        19.4     Stack
   C          14         4.3     Code Segment from CST
   P          52         8.3     Code Segment from Program File
   T           4          .3     System Table Data Segment
   F          71         3.1     File System Data Segment
   J          11          .1     Job Management Data Segment
   I          47         1.9     Image Data Segments
   K           4         2.4     KSAM Data Segments
   D          50         2.9     Data Segment (usage unknown)
   A         301        42.6     Active Segment
   *           0          .0     Segment on Overlay Candidate List
   /          12         2.5     Segment that is Frozen, Locked or I/O Frozen

% OF MEMORY ACTIVE IN EACH BANK:
   BANK  0:100.0   BANK  1: 99.6   BANK  2: 93.4   BANK  3:   .0   BANK  4: 97.5   BANK  5: 96.7   BANK  6: 93.0   BANK  7: 97.1
   BANK 10: 97.7   BANK 11: 99.0   BANK 12: 70.9   BANK 13: 96.1   BANK 14: 99.2   BANK 15: 77.7   BANK 16: 99.2   BANK 17:100.0
```

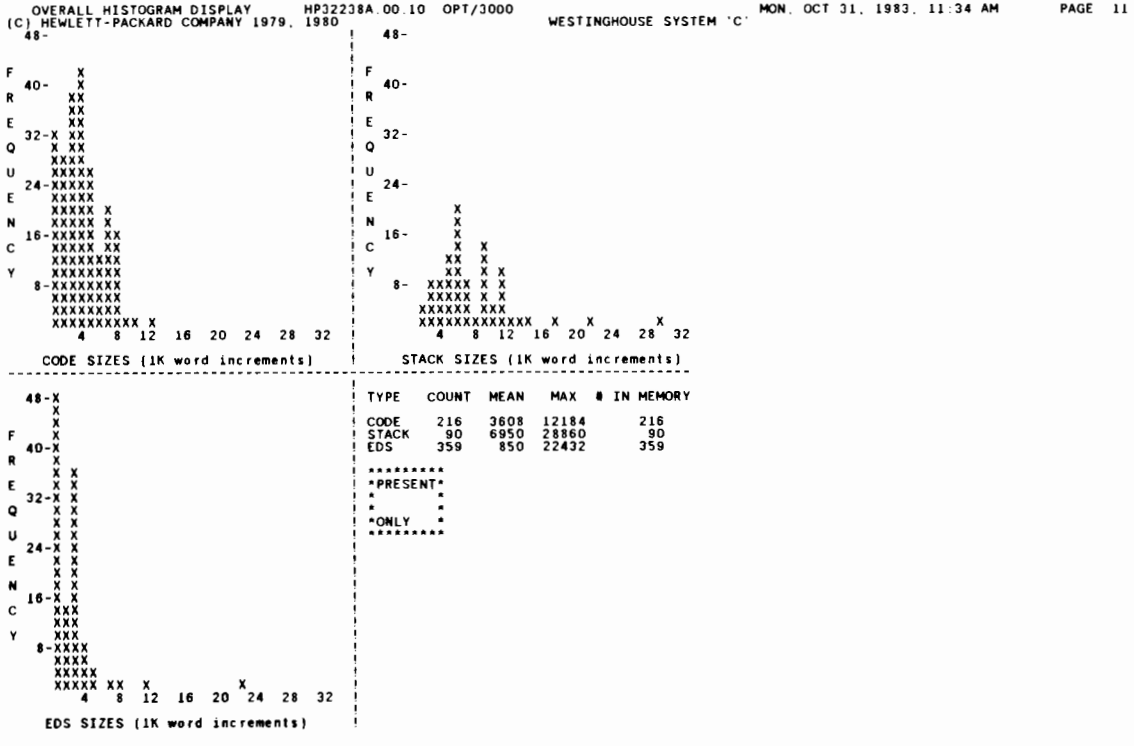
BANK CONTENTS DISPLAY HP32238A.00.10 OPT/3000 WESTINGHOUSE SYSTEM 'C' MON, OCT 31, 1983, 11:39 AM PAGE 15
(C) HEWLETT-PACKARD COMPANY 1979, 1980
BANK NUMBER 5

BANK CONTENTS DISPLAY HP32238A.00.10 OPT/3000 WESTINGHOUSE SYSTEM 'C' MON, OCT 31, 1983, 11:39 AM PAGE 16
(C) HEWLETT-PACKARD COMPANY 1978, 1980

LEGEND AND SUMMARY STATISTICS

Each position represents 64 words of memory. The starting and ending address of the segment are rounded to the nearest 64 word increment, so that the numbers above and to the left of the display specify the starting address of each segment.
Lower case denotes a system segment

SYMBOL	SEGMENT COUNT	% OF LINKED MEMORY	DESCRIPTION
S	2	23.2	Stack
C	2	23.6	Code Segment from CST
P	6	22.1	Code Segment from Program File
T	0	.0	System Table Data Segment
F	11	9.2	File System Data Segment
J	1	.2	Job Management Data Segment
I	11	8.4	Image Data Segments
K	0	.0	KSAM Data Segments
D	4	7.8	Data Segment (usage unknown)
A	37	87.5	Active Segment
*	0	.0	Segment on Overlay Candidate List
/	0	.0	Segment that is Frozen, Locked or I/O Frozen
X	4	12.5	Free Area


OVERALL HISTOGRAM DISPLAY HP32238A.00.10 OPT/3000 MON, OCT 31, 1983, 11:34 AM PAGE 11
(C) HEWLETT-PACKARD COMPANY 1979, 1980 WESTINGHOUSE SYSTEM 'C'
FREQUENCY
CODE SIZES (1K word increments)
STACK SIZES (1K word increments)
EDS SIZES (1K word increments)
TYPE COUNT MEAN MAX # IN MEMORY
CODE 216 3608 12184 216
STACK 90 6950 28860 90
EDS 359 850 22432 359
PRESENT
ONLY

```
          CPU USAGE DISPLAY      HP32238A.00.10  OPT/3000                    MON, OCT 31, 1983, 11:31 AM       PAGE   3
(C) HEWLETT-PACKARD COMPANY 1978, 1980                    WESTINGHOUSE SYSTEM 'C'
   CURRENT INTERVAL:     11.7 SECONDS          OVERALL INTERVAL:     1.3 MINUTES

                        10        20        30        40        50        60        70        80        90       100%
CURRENT CPU STATE B----------------------------------------BPPD----------------------------------------DO----------O (46/2/38/14)
OVERALL CPU STATE B----------------------------------------BPPD----------------------------------------DO----------O (52/2/30/16)

                        10        20        30        40        50        60        70        80        90       100 per second
                  ----------------------------------------------------------------------------------------
  PROCESS LAUNCHES !                                                                        !/////////!            (77/86)
                  ----------------------------------------------------------------------------------------
                  ----------
  PROCESS PREEMPTS !        *!!                                                                               (19/20)
                  ----------

CPU STATE INFORMATION
                    AVG TIME IN STATE(ms)   STATE PERCENTAGES          AVERAGE CPU
                      CURRENT    OVERALL    CURRENT    OVERALL         TIME PER
                                                                       TRANSACTION
          CPU BUSY          8          6      46.2%      50.0%
PAUSED DISC & SWAP         22         22       1.5%       2.0%       OF INTERACTIVE
   PAUSED FOR DISC         20         19      37.3%      30.7%       PROCESSES
  PAUSED FOR SWAP                             .0%        .0%
              IDLE                            .0%        .0%       CURRENT:  116 MS
GARBAGE COLLECTION                            .0%        .0%       MAXIMUM:  267 MS
 MEMORY ALLOCATION          3          3       .2%        .2%


LEGEND FOR PROCESS LAUNCHES AND PREEMPTS:

---               ---               ---
! ! CURRENT RATE  !/! MAXIMUM RATE  !*! OVERALL RATE
---               ---               ---

CPU STATE LEGEND:

  B  Busy on process
  P  Paused for disc and swap
  D  Paused for disc
  S  Paused for swap
  I  Idle
  G  Garbage collection
  M  Memory allocation
  O  ICS overhead
```

```
      MAM ACTIVITY DISPLAY      HP32238A 00 10  OPT/3000                    MON, OCT 31, 1983, 11:31 AM       PAGE   4
(C) HEWLETT-PACKARD COMPANY 1979, 1980          WESTINGHOUSE SYSTEM 'C'
   CURRENT INTERVAL:      6.0 SECONDS          OVERALL INTERVAL:  &a01r55C dONDS

                          10        20        30        40        50        60        70        80        90       100%
CURRENT MM ACTION F-------------------------------------------------------------------------------------------F        (6)
OVERALL MM ACTION RRF-----------------------------------------------------------------------------------------F       (67)

                          10        20        30        40        50        60        70        80        90       100 per second
                          --
MEMORY ALLOCATIONS       *!                                                                                          (1/1)
                          --
                          --
  PROCESS SWAP-INS       *!                                                                                          (1/1)
                          --


                          --
     MM I/O  READS       *!                                                                                          (1/1)
                          --
                          --
     MM I/O WRITES        !                                                                                          (1/1)
                          --


  RELEASE DATA SEG

  RELEASE CODE SEG

SPECIAL REQUESTS-  CURRENT:    67/second        CLOCK CYCLE RATE- CURRENT:     .00/second
PERCENT:   25.84%  OVERALL:    .25/second                         OVERALL:     .00/second

LEGEND FOR EVENT RATES:

 ---               ---               ---
 ! ! CURRENT RATE  !/! MAXIMUM RATE  !*! OVERALL RATE
 ---               ---               ---

MM ACTION LEGEND:

  R  Recovered overlay candidate
  F  Found free region
  O  Made overlay candidates
  G  Give up
  H  Hard request
```

```
     CPU-MAM REPORT DISPLAY      HP32238A.00.10  OPT/3000                    MON, OCT 31, 1983, 11:31 AM       PAGE   5
(C) HEWLETT-PACKARD COMPANY 1979, 1980                  WESTINGHOUSE SYSTEM 'C'
   CURRENT INTERVAL:     5.8 SECONDS          OVERALL INTERVAL:     1.8 MINUTES

                        10        20        30        40        50        60        70        80        90       100 per second
                    ----------------------------------------------------------------------------------------------------
  PROCESS LAUNCHES  !                                                                                !//////////////////!     (76/94)
                    ----------------------------------------------------------------------------------------------------
                    --
  PROCESS SWAP-INS  *!                                                                                                     (1/1)
                    --


                    --
MEMORY ALLOCATIONS  *!                                                                                                     (1/1)
                    --

                        10        20        30        40        50        60        70        80        90       100%
 CURRENT CPU STATE B----------------------------------------------BD----------------------DO----------------O (54/28/18)
 OVERALL CPU STATE B----------------------------------------------BPPD--------------------DO----------------O (50/2/30/18)

 CURRENT MM ACTION F-------------------------------------------------------------------------------------------F        (2)
 OVERALL MM ACTION RRF-----------------------------------------------------------------------------------------F       (72)

LEGEND FOR LAUNCHES, SWAP-INS, AND MEMORY ALLOCATIONS:

---   ---   ---
! ! CURRENT RATE    !/! MAXIMUM RATE    !*! OVERALL RATE
---   ---   ---

CPU STATE LEGEND:

  B  Busy on process
  P  Paused for disc and swap
  D  Paused for disc
  S  Paused for swap
  I  Idle
  G  Garbage collection
  M  Memory allocation
  O  ICS overhead

MM ACTION LEGEND:
  R  Recovered overlay candidate
  F  Found free region
  O  Made overlay candidates
  G  Give up
  H  Hard request
```

```
 I/O ACTIVITY REPORT DISPLAY     HP32238A.00.10  OPT/3000     WESTINGHOUSE SYSTEM 'C'  MON, OCT 31, 1983, 11:32 AM     PAGE  6
(C) HEWLETT-PACKARD COMPANY 1979, 1980          OVERALL INTERVAL:     2 MINUTES
   CURRENT INTERVAL:    13.5 SECONDS

ACTIVITY IN CURRENT INTERVAL

                10        20        30        40        50        60        70        80        90        100 per second
                |---------|---------|---------|---------|---------U         |         |         |         (54)
DISC I/O ACTIVITY U
TAPE I/O ACTIVITY
PRINTER ACTIVITY

ACTIVITY OVER ALL INTERVALS

                10        20        30        40        50        60        70        80        90        100 per second
                |---------|---------|---------|---------|---------U         |         |         |         (54)
DISC I/O ACTIVITY U
TAPE I/O ACTIVITY
PRINTER ACTIVITY


LEGEND FOR DISC I/O ACTIVITY:
  U  User disc I/O
  M  Memory management disc I/O
```

```
     DISC ACTIVITY DISPLAY    HP32238A.00.10  OPT/3000     WESTINGHOUSE SYSTEM 'C'   MON, OCT 31, 1983, 11:33 AM      PAGE  10
(C) HEWLETT-PACKARD COMPANY 1979, 1980
  CURRENT INTERVAL:  4.3 SECONDS          OVERALL INTERVAL:  1.2 MINUTES

                    10        20        30        40        50        60        70        80        90       100 per second
CURRENT ALL DISC  R-------------------------------------------------------R                                  (66)
OVERALL ALL DISC  R-------------------------------------------------------R                                  (44/10)

  CURRENT LDEV  1 R-------------------------RW-------W                                                      (30)
  OVERALL LDEV  1 R------R                                                                                 (8)

  CURRENT LDEV  2
  OVERALL LDEV  2

  CURRENT LDEV  3
  OVERALL LDEV  3 RRWW                                                                                     (2/2)

  CURRENT LDEV  4 R-------------------------R
  OVERALL LDEV  4 R-------------------------RWW                                                             (34)
                                                                                                           (26/2)
-------------------------------------------------------------------------------------------------------------------

MAXIMUM RATES:  ALL   LDEV  1   LDEV  2   LDEV  3   LDEV  4
                ---   ------    ------    ------    ------
          READS  33/ 1   15/ 0     1/ 0     4/ 1     22/ 0
         WRITES   8/ 1    1/ 0     1/ 0     1/ 1      2/ 0
        CONTROL   1 1     0        0        0         1


LEGEND FOR I/O OPERATION RATES:
  R  Read operation
  W  Write operation
  C  Control operation

MAXIMUM RATES FOR READ AND WRITE OPERATIONS ARE SHOWN IN THE FORM 'USER I/O / MEMORY MANAGEMENT I/O'
```

```
        PROGRAM FILE DISPLAY       HP32238A.00.10  OPT/3000                                 MON, OCT 31, 1983, 11:43 AM       PAGE  21
(C) HEWLETT-PACKARD COMPANY 1979, 1980                         WESTINGHOUSE SYSTEM 'C'
PROGRAM FILE DISPLAY                                                        11:43 AM
                                 CODE SEG INFO        COMBINED WORKING SET INFORMATION
   NAME OF PROGRAM FILE           COUNT:SIZE   #PS     CST:PROG: OST:STACK:TOTAL SIZE
---------------------------------------------------------------------------------------
SP SYSTEM PROCESSES                              21      6 :  4 : 40 : 18  :  194180
CI COMMAND INTERPRETER                            7      0 :  0 : 23 :  7  :   71244
 1 DSMON    .PUB     .SYS         3:  3484        2      1 :  0 : 10 :  2  :   45300
 2*GRM0003P.PRODUCTN.MARKETNG     2:  7344        9      0 :  0 : 16 :  9  :  113068
 3 MPMON    .PUB     .SYS         8: 23004        3      0 :  0 : 14 :  3  :   55616
 4 LRISP11P.PRODUCTN.MARKETNG     5: 15332        1      0 :  0 :  6 :  1  :   36920
 5 GRSC060P.PRODUCTN.MARKETNG     8: 20784        1      0 :  0 :  6 :  1  :   39912
 6 LRISP09P.PRODUCTN.MARKETNG     1:  4916        1      0 :  0 :  7 :  1  :   36296
 7 LRISP71P.PRODUCTN.MARKETNG     2: 14116       13      3 :  1 : 35 : 13  :  191516
 8 LRISP70P.PRODUCTN.MARKETNG     1:  1748        1      0 :  0 :  2 :  1  :    4044
 9 SAMPLER .PUB     .SYS          6: 14488        1      0 :  0 :  2 :  1  :    7464
10 LRISP08P.PRODUCTN.MARKETNG     9:  9604       10      0 :  3 : 32 : 10  :  108552
11 User Program File              8: 10136        1      0 :  0 :  1 :  1  :   16104
12 OPT     .PUB     .SYS         36:125664        1      0 :  4 :  3 :  1  :   24344
13 P03P180A.PUB     .SYS          1:   448        1      0 :  1 :  3 :  1  :   11392
14 GRSC050P.PRODUCTN.MARKETNG    11: 17928        1      0 :  0 :  5 :  1  :   20416
15 User Program File              4: 11796        1      0 :  0 :  3 :  1  :   21296
16 LRISP07P.PRODUCTN.MARKETNG     4:  6252        1      0 :  0 :  4 :  1  :    8348
```

USER SUMMARY DISPLAY HP32238A.00.10 OPT/3000 WESTINGHOUSE SYSTEM 'C' MON, OCT 31, 1983, 11:42 AM PAGE 20
(C) HEWLETT-PACKARD COMPANY 1979, 1980
USER SUMMARY REPORT

PIN	USER.ACCT	PROGRAM NAME (command)	CPU	%	PRI	WORKING SET CSTSZ	WORKING SET STKSZ	INFO DSTSZ
11	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	60784	0	152	0	8948	34140
17	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	21356	2	152	1932	8948	3880
20	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	37452	0	152	0	8948	28064
21	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	14711	0	152	10858	8948	25752
29	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	18984	2	152	0	8948	724
30	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	25188	2	152	0	8948	724
31	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	24108	0	152	0	8948	5928
32	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	34302	0	152	0	8948	724
33	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	47002	0	152	4392	8948	772
34	MANAGER.SYS	user program file	221	0	149	448	2568	8376
36	PILOT.MARKETNG	GRSC050P.PRODUCTN.MARKETNG	1482	0	102	0	11864	8452
39	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	23243	0	152	0	6080	18032
41	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	1838	0	156	0	2804	33492
42	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	30732	0	152	0	6080	7760
43	MANAGER.SYS	LRISP08P.PRODUCTN.MARKETNG	7592	0	152	0	6080	160
45	PILOT.MARKETNG	user program file	61	0	150	0	4072	3712
46	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	8525	1	152	0	6080	4824
47	PILOT.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	55160	2	152	0	10388	4420
48	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	25007	2	152	0	6080	1638
50	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	177	0	152	3080	4200	3712
51	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	6914	0	152	0	8948	5944
53	USER.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	1134	0	152	0	11872	4628
54	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	52783	0	152	0	6080	18344
55	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	181	0	152	3080	4200	3712
56	PILOT.MARKETNG	LRISP08P.PRODUCTN.MARKETNG	131175	2	184	0	6080	9360
58	USER.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	2004	0	152	0	10388	4392
60	USER.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	2004	0	152	0	10388	4392
61	USER.MARKETNG	GRSC140P.PRODUCTN.MARKETNG	480	1	152	2724	5168	3710
63	USER.MARKETNG	GRSC060P.PRODUCTN.MARKETNG	179853	0	152	0	21156	18756
65	PILOT.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	1729	0	152	0	10388	4420
72	USER.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	2580	0	152	0	10388	4260
76	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	29103	0	152	0	8948	5944
82	PILOT.MARKETNG	LRISP07P.PRODUCTN.MARKETNG	9261	2	152	0	7004	1344
94	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	13318	0	152	0	8948	5992
99	MANUFACT.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	16455	0	152	0	10388	13208
102	MANAGER.SYS	user program file	2794	0	152	0	6080	4124
103	PILOT.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	2794	0	152	0	10388	4420
107	USER.MARKETNG	GRMO003P.PRODUCTN.MARKETNG	1011	0	152	0	10388	4260
108	MANAGER.MARKETNG	OPT.PUB.SYS	181478	4	200	8116	12748	1178
111	SHPOUT.SHPLOCTR	user program file	744	0	152	0	14924	1180
115	PILOT.MARKETNG	LRISP11P.PRODUCTN.MARKETNG	183089	4	152	0	28860	8060
118	SHPIN.SHPLOCTR	user program file	50084	2	152	0	13720	7576
120	USER.MARKETNG	LRISP70P.PRODUCTN.MARKETNG	3480	0	158	0	2724	1320
127	USER.MARKETNG	LRISP71P.PRODUCTN.MARKETNG	26308	0	152	0	8948	28064

```
PROCESS STATE REPORT DISPLAY      HP32238A.00.10  OPT/3000                              MON, OCT 31, 1983, 11:44 AM       PAGE  23
(C) HEWLETT-PACKARD COMPANY 1979, 1980                    WESTINGHOUSE SYSTEM 'C'
 SESSIONS:  23            SYSTEM PROCESSES:  21   DISPATCH QUEUE    4
     JOBS:   3                CI PROCESSES:   6         NO QUEUE   87
                         USER PROCESSES:  20
 IN BREAK:   0          CREATED PROCESSES:  24

-------------------------------------------------------------------------------
                    DISTRIBUTION OF PROCESS STATES

                  DISPATCH QUEUE     NO QUEUE
                  --------------     --------
      SHORT WAIT          1
       LONG WAIT          1              71
  TERM READ WAIT          1              40
BLOCKED I/O WAIT                         40
        I/O WAIT                          5

     MEMORY WAIT                          1
        SON WAIT                         25
     FATHER WAIT                          7
        RIN WAIT
        SIR WAIT
      IMPEDED                            4
  SCHED ATTN REQ
```

```
                    PROCESS DISPLAY      HP32238A.00.10  OPT/3000        WESTINGHOUSE SYSTEM 'C'  MON, OCT 31, 1983, 11:48 AM      PAGE  27
(C) HEWLETT-PACKARD COMPANY 1979, 1980
PIN:  11  LRISP71P.PRODUCTN.MARKETNG  USER: USER.MARKETNG      (J  17)  11:48 AM
-------------------------------------------------------------------------------------------------------------------------------
     STACK INFORMATION      1185 13.4%  |   CPU TIME:  60811 MSEC  | STATUS FLAGS:
       DST:  263* SYSQ:       32  1.8%  |                          |
      SIZE:  8948 DB-Q:     6083 67.8%  |   PRIORITY: 152          |   BIO
   MAXDATA:  9581 DB-QD:     398  4.4%  |                          |
MAX Z-DL.:  7753   Q1-Q:       6  .1%  |   CAP:                   |
                  Q-S:    1194 13.3%  |     LG ND SF             |
                  S-Z:                |     IA BA                |
-------------------------------------------------------------------------------------------------------------------------------

XDS USAGE          SON
DST#  SIZE   PROCESS   OPEN FILES
-----------  -------   ----------
  210* 5448            $STDIN    $STDLIST SCOEDB    LRISCT    ISR23A02 LRISD16P
  215* 22440           $STDIN    $STDLIST LRISD17P  DCINWM02  ISR23A05 LRISC01
  337* 252             SCOEDB10  LRISC12  SCOEDB06  LRISCT05  LRISCT04  LRISC09  LRISC01
  307* 10928           SCOEDB11  SCOEDB05 LRISCT07  LRISCT06  SCOEDB04  SCOEDB02
  352* 548             SCOEDB13  SCOEDB12
  215* 22440

CST #  SIZE          CSTX #  SIZE          DST #   SIZE
-----------         -------  ----          -----   ----
                                           307*  10928
                                           215*  22440
                                           514*    772
                                           263*S  8948

STACK MARKER INFORMATION:

ADDRESS   Q-7    Q-6    Q-5    Q-4     X   DELTAP STATUS DELTAQ SEGMENT-NAME
-------   ---    ---    ---    ---     -   ------ ------ ------ ------------

014475 000283 000000 000200 000001 001053 015604 101014 000017 HARDRES
014458 000000 000040 000020 003771 027555 014682 140011 000037 HARDRES
014423 177634 000001 000000 120001 000144 006320 140473 000183 FILESYS1A
014240 000000 000020 013737 177634 000004 002153 142473 000112 FILESYS1A
014128 000000 000012 013737 177634 000003 001733 060622 000010 DCAINTR
014118 013737 013726 013723 013725 000003 007375 060222 000174 DCAINTR
013722 004214 004221 004223 004215 000172 026085 060301 000037 (user segment)
013663 016017 000000 000000 000000 000000 000000 140105 000004 MORGUE
```

TABLE INFORMATION DISPLAY HP32238A.00.10 OPT/3000 MON, OCT 31, 1983, 11:40 AM PAGE 17
(C) HEWLETT-PACKARD COMPANY 1979, 1980 WESTINGHOUSE SYSTEM 'C'

	CONFIGURED ENTRIES	CURRENT USAGE IN USE	CURRENT USAGE UTIL	MAXIMUM USAGE OBSVD	MAXIMUM USAGE IN USE	MAXIMUM USAGE UTIL	ENTRY SIZE	
CODE SEGMENT TABLE	192	149	77.6%	149	150	78.1%	4	W
EXTENDED CST	512	173	33.8%	173	187	36.5%	4	W
DATA SEGMENT TABLE	1024	497	48.5%	497	510	49.8%	4	W
PROCESS TABLE	128	92	71.9%	92	97	75.8%	16	W
I/O QUEUE TABLE	160	52	32.5%	52	72	45.0%	11	W
DISC REQUEST TABLE	120	4	3.3%	4	14	11.7%	16	W
TERMINAL BUFFERS	255	32	12.5%	32	54	21.2%	32	W
ATP TERM BUFFERS								
SYSTEM BUFFERS	24		%		1	4.2%	129	W
SWAP TABLE	750	549	73.2%	549	549	73.2%	5	W
CST BLOCK TABLE	58	27	46.6%	27	28	48.3%	1	W
PRIMARY MSG TABLE	25	1	4.0%	1	1	4.0%	5	W
SECONDARY MSG TBLE	25	1	4.0%	1	1	4.0%	5	W
SPECIAL ROST TABLE	25	1	4.0%	1	1	4.0%	5	W
INTERRUPT CTRL STK	1024		%		375	36.6%	1	W
UCOP REQUEST QUEUE	48		%		0	0%	2	W
TIMER REQUEST LIST	60	24	40.0%	24	33	55.0%	4	W
BREAK POINT TABLE	80		%		0	0%	VAR	
RIN TABLE	64	28	43.8%	28	28	43.8%	2	W
JOB PROC CNT TABLE	45	26	57.8%	26	26	57.8%	1	B
VIRTUAL MEMORY	30	18	60.0%	18	19	63.3%	1024	S
SPOOLER DISC SPACE	100	5	5.0%	5	5	5.0%	1000	S

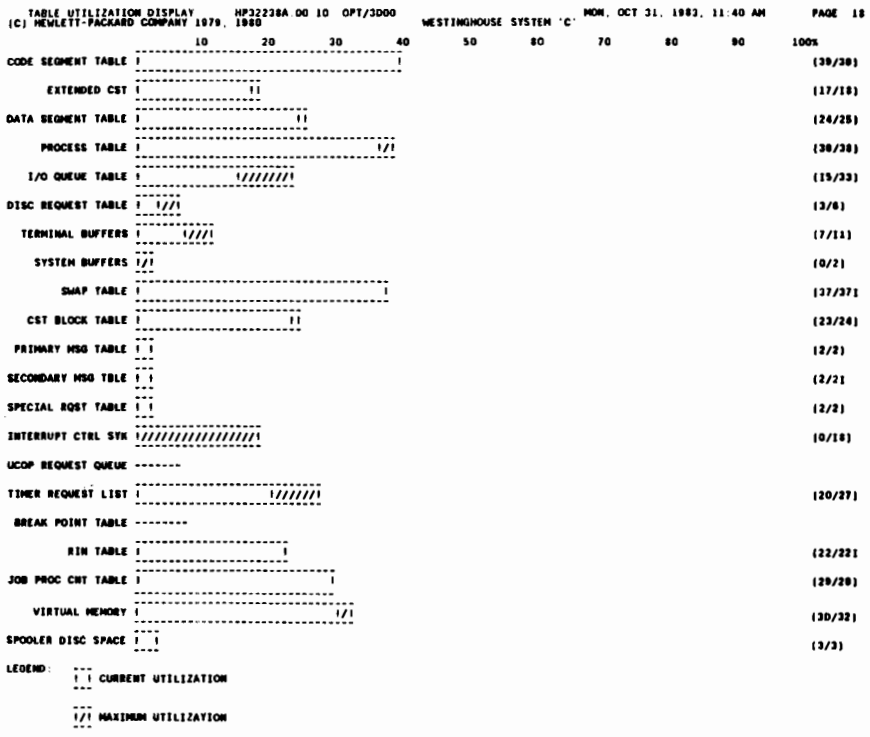


TABLE UTILIZATION DISPLAY HP32238A.00.10 OPT/3000 MON, OCT 31, 1983, 11:40 AM PAGE 18
(C) HEWLETT-PACKARD COMPANY 1979, 1980 WESTINGHOUSE SYSTEM 'C'
10 20 30 40 50 60 70 80 90 100%
CODE SEGMENT TABLE (30/30)
EXTENDED CST (17/18)
DATA SEGMENT TABLE (24/25)
PROCESS TABLE (30/30)
I/O QUEUE TABLE (15/33)
DISC REQUEST TABLE (3/6)
TERMINAL BUFFERS (7/11)
SYSTEM BUFFERS (0/2)
SWAP TABLE (37/37)
CST BLOCK TABLE (23/24)
PRIMARY MSG TABLE (2/2)
SECONDARY MSG TBLE (2/2)
SPECIAL RQST TABLE (2/2)
INTERRUPT CTRL SYK (0/16)
UCOP REQUEST QUEUE
TIMER REQUEST LIST (20/27)
BREAK POINT TABLE
RIN TABLE (22/22)
JOB PROC CNT TABLE (20/20)
VIRTUAL MEMORY (30/32)
SPOOLER DISC SPACE (3/3)
LEGEND:
CURRENT UTILIZATION
MAXIMUM UTILIZATION

RUNNING A SMALL-SHOP INFORMATION SYSTEM BASED ON THE USER/PROGRAMMER CONCEPT

Eric W. Roberts
Information Resource Manager
The Audichron Company

INTRODUCTION

The purpose of this paper is to show how an Information System can be set-up and run successfully at minimum cost in a small to medium-sized company. The ideas presented can be used either by a company with an existing DP shop, or by a company just starting one up. The user/programmer concept is based on use of the following resources:

- Purchased, integrated applications software.
- Fourth generation languages.
- User/programmers who write their own report programs.
- Departmental Computer Coordinators.

We will take a look at the three main ingredients of the system: software, hardware, and people, and then outline some helpful rules of the road that help to maintain control in this unique environment.

SOFTWARE

Start off by purchasing a good family of integrated applications software packages, all from the same vendor. Don't try to buy an MRP system from one vendor and try to glue it to someone else's order entry package. Take plenty of time to evaluate several packages, and make sure the evaluation includes at least one hands-on session. The most important criteria, once you get past the price tag, is ease of use. Other important factors are vendor technical support and the life expectancy of the vendor.

Next on your shopping list is a package of fourth generation language (4GL) products which should include as a minimum a report generator, a screen-oriented data entry and lookup tool, and a data dictionary. Shops with high volume transactions should consider a batch data entry product. Select your 4GL products from the same vendor to make it easy on yourself when it comes time to train your users.

There is no family of applications software that is going to meet ALL of your users' needs, so you will have to develop a few small applications in-house, preferably using IMAGE databases along with your 4GL toolkit. Rely on your users to develop these applications for you; more on that in the "PEOPLE" section later.

Round out your software inventory with a couple of purchased housekeeping utilities like ADAGER to pull maintenance on your databases, and MPEX to keep your discs tidy. Packages like these are inexpensive, and will pay for themselves quickly.

Finally, you should purchase software maintenance agreements from each of your vendors. You will not have the time nor the people required to maintain your purchased packages in-house.

Figure 1 illustrates how users in a typical small-shop company access the databases using a combination of 4GL products and off-the-shelf software. The example portrays the 4GL products of Quasar Systems, Ltd., and the MANMAN family of applications software from ASK Computer Systems, Inc.

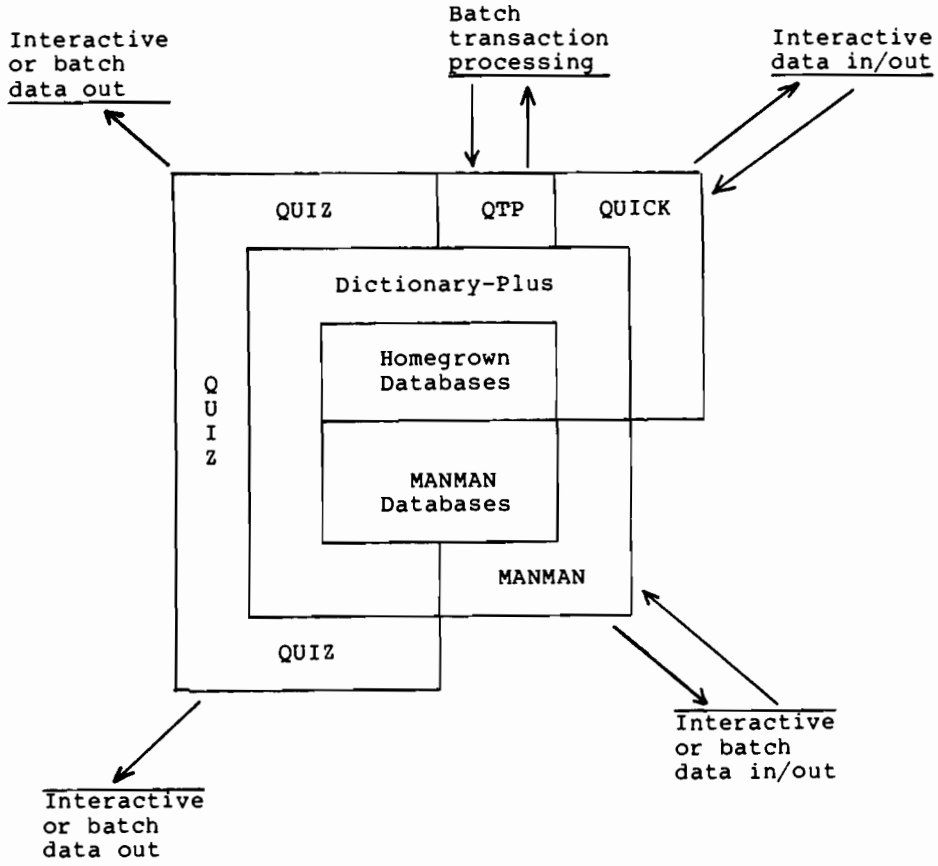


Figure 1
Example of Combination of Purchased and Homegrown Databases
and Methods of Access

HARDWARE

Buy only as much mainframe hardware as you need now, plus what you foresee within a horizon of one year. Don't buy a 68 now and wait three years only to find out that you could have gotten by with a 48. The architecture of the HP3000 lends itself very nicely to future expansion, since HP has pursued, and hopefully will continue to pursue, a policy of upward compatibility in both hardware and operating system.

The same principle applies to peripherals such as line printers, tape drives, and disc drives. Buy enough disc space to let you run at not more than 80% capacity over the next year. It's easy enough to add on extra capacity when the time comes. With a small-shop IS, whole idea is to keep every aspect of your system as simple as possible to minimize workload on your staff, and dollars in your budget.

If your users are far-flung and you have to get into communications gear such as multiplexers and modems, be sure to buy equipment that is reliable and guaranteed compatible with the HP3000. You will have enough problems with the computer itself and the people using it without having to be bothered with data communications failures.

In the area of terminals, purchase only high-reliability CRT's and printing terminals. The extra reliability will cost extra dollars, but you'll be thankful you made the additional outlay up front. Unreliable terminals cause user frustration and higher maintenance bills, not to mention the time it takes your tiny staff to do the initial troubleshooting.

Let each department purchase their own terminals, but give them a list of approved terminals from which to choose. If you buy their terminals for them, you'll spend lots of time chasing down promise dates from the vendor every time a user calls you to ask "Where's my terminal?". Also, if a department owns a terminal, they're much more likely to be conscientious about taking care of it. The System Manager must rule over the allocation of new ports with an iron fist. Don't let the idea take hold amongst your users that terminals are like telephones, and one belongs on every desk. Try to make sure that terminals are placed in common areas where multiple users can access them, not on a secretary's desk where she is a dedicated user. Dedicated terminals should be placed only at the desks of heavy-duty data entry people, such as stock clerks and order takers.

Some of your users will be quick to try to talk you into buying some microcomputers, since they are the fad now, and all the trade magazines have pictures of executives with their micros on their credenzas. Beware the micro trap! Install micros ONLY where they are needed for very specialized applications that the HP3000 doesn't handle well. If you do install micros, assure that they are 100% capable of communicating with the HP3000, to include uploading and downloading of files. The HP120, HP125, and HP150 micros are good examples of machines with total HP3000 compatibility.

PEOPLE

The first "people" thing that has to happen is the formation of a small IS staff. An existing company with a large IS staff will have to trim down some excess fat if they are to adopt the user/programmer concept. Large shops are going to tend to want to justify their continued existence, so they will probably demand that the purchased packages be modified so they'll more closely resemble the in-house system that is being phased out. Top management will have to be aware of this natural tendency of self-preservation, and act early to counter it when organizing for a small shop.

Figure 2 shows a typical placement of the IS function within a small to medium-sized business. It is important that the Information Resource (IR) department not answer to one of the using department managers, so as to assure objectivity of the services rendered by IR personnel. It is also important that the Information Resource Manager (IRM) be placed on the same level as other department managers to assure that he will have sufficient clout to implement IS plans and policies company-wide. In a medium-sized company there may even be a Vice-President for Information Resources answering directly to the President, which is the ideal situation in terms of implementing an integrated corporate Information System.

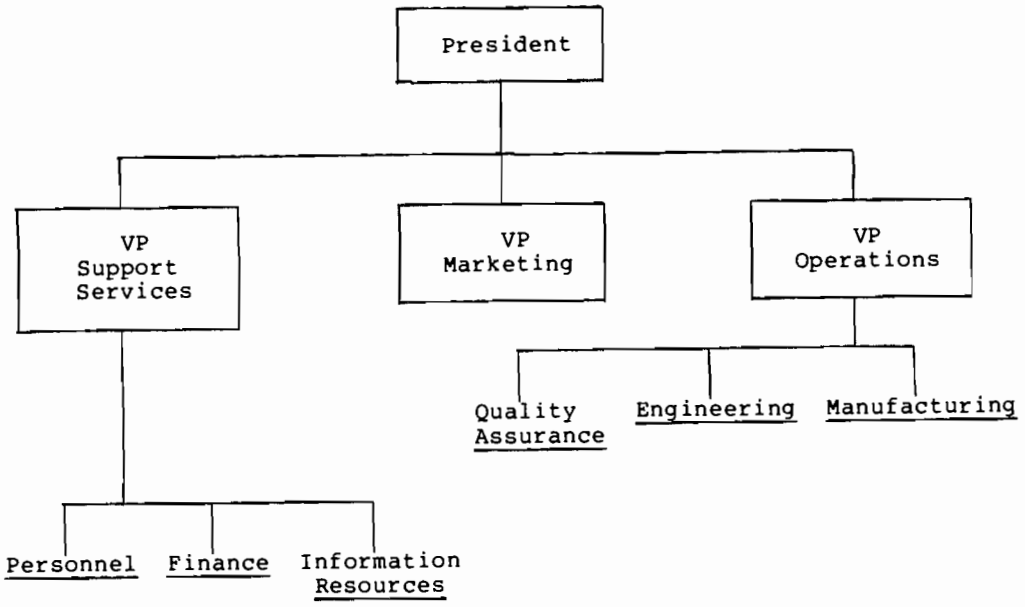


Figure 2
Typical Placement of Information Resources in the Organization Chart

Figure 3 shows the bare minimum IR staff required to get the job done with the user/programmer concept. Some larger companies may need to change the Programmer/Analyst slot to Systems Analyst, and put one or two Programmers under the Systems Analyst. It would be nice to have a full-time department secretary to help the System Administrator with documentation and correspondence. In any case, you get the point: keep the staff trimmed down to the bare minimum required. Don't fall into the trap of thinking that a company with X employees should have an IR staff of size Y, replete with the traditional cubicles of Programmers and Analysts. That is the old in-house MIS way of thinking. The new Information Resource concept, based on the user/programmer, is a relatively new way of thinking in the field of data processing.

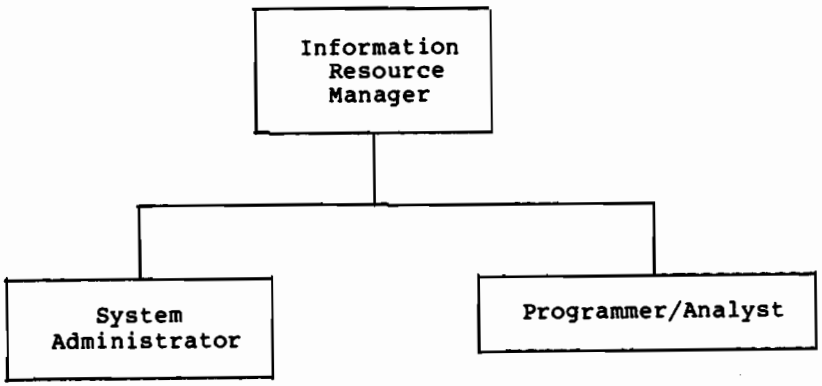


Figure 3
Organization of a Small Information Resource Department

Following are abbreviated job descriptions for the IR staff:

Information Resource Manager:

- Serve as System Manager for the HP3000.
- Coordinate all off-line information systems such as word processing, TWX/TELEX, and microcomputers.
- Publish and maintain the Company Procedures Manual.
- Conduct all in-house training for use of the HP3000.

System Administrator:

- Operate the HP3000 computer, with System Manager capability.
- Create and purge accounts, groups, and users.
- Maintain all departmental records.
- Manage the tape archives and Contributed Software Libraries.
- Publish usage reports and traffic studies.
- Place service calls to HP Customer Engineering.

Programmer/Analyst:

- Develop and maintain major in-house applications.
- Act on users' Requests for Service.

- Write system utilities as directed by the IRM.

- Install patches as directed by vendor technical personnel.

So let's assume that the staff shown in Figure 3 is installed and we're ready to proceed. The next "people" ingredient that is needed is a Steering Committee, composed of top management plus the Information Resource Manager. The President, or the person to whom the IRM reports, whichever is higher, should chair the committee. The IRM should NEVER chair the committee, so as to maintain a system of checks and balances. The committee should also contain at least one employee who is technically savvy regarding the HP3000, and who does not work in the Information Resource department. This accomplish two objectives. First, it gives the committee another angle of expertise when technical issues arise. Secondly, since the IR staff usually works in a tight concentration around the computer room, a major catastrophe could take out the entire pool of Information Systems expertise in one fell swoop. At least one person physically removed from the IR department should be kept continually in the know as to the daily operations of the HP3000, and should also sit on the Steering Committee.

The Steering Committee should meet monthly or quarterly and develop and maintain long-range plans and goals for the corporate Information System. The IRM should not report to the committee, since committees usually make for lousy bosses, but he should be receptive to the committee's input. Conflicts should be resolved by the chairman, who is usually the IRM's supervisor anyway. Since the IRM will be doing most of the reporting at the meetings, he should serve as the committee's secretary.

Training is a key facet of the user/programmer concept. A lot of people have to be trained in a lot of different areas. Before laying out the training plan, the idea of the Alternate System Manager and the Departmental Computer Coordinator has to be explained.

Once the staff is in place, the IRM must select certain individuals to serve as Alternate System Managers (ASM's). An ASM is one who would take over System Management duties in the event of the IRM's temporary or (heaven forbid) permanent absence. Top management must assure that the IRM is not allowed to be the only employee carrying all the information eggs in his basket. Some ASM's must be trained to take over in case the proverbial truck catches up with the IRM, so that the Information System doesn't ground to a halt within a few days of the unfortunate mishap. The IRM and anyone who is designated to be an ASM should be sent either to Programmer's Intro or System Operator school at HP, and then to the System Manager course. If your applications software vendor offers a System Manager course, by all means send them to that course also. Then the IRM should conduct an in-house course to supplement the generic courses by teaching policies and procedures peculiar to the company. The System Administrator and the Programmer/Analyst plus the one non-IS type person mentioned above should all be trained as ASM's as a minimum. Three alternates should suffice in most companies. Any more or less would probably make your EDP auditors a little nervous.

Next, the IRM and all of the ASM's should attend training courses given by the vendors on applications software and 4GL products. These courses typically go into great detail on the structure of each package and how to use them for maximum gain. Armed with this knowledge, the IRM and his charges should return home and set-up training courses for all of the users. The IRM should plan on teaching all of these courses himself, perhaps with some help from the ASM's. A training room with a few terminals is essential to let the users get hands-on training. The user classes should not be intensive like the vendor's, but should be tailored down so as to give the users in your particular company only the knowledge needed to get their particular jobs done. All users should attend as a minimum classes on the applications software, and then classes on the 4GL products, if they show an interest in programming.

Now the IRM must put together a "staff" of Departmental Computer Coordinators (DCC's), all of whom should have gone through the appropriate set of user training classes. These DCC's will form the cornerstones of your small-shop IS concept. They will be the first line of contact for your users, answering questions, helping with terminal problems, and helping to write 4GL programs. You should train at least two DCC's in each using department, one of which should be the department manager. The department manager may never logon as a bonafide user, but at least you will know that he has been exposed to what the computer can and cannot do. The manager will also be better equipped to survive meetings at which computerese is being bandied about the table.

The second DCC for any given department should be a person who has some experience with computers, is comfortable around them, and wouldn't mind helping out his fellow

workers with computer applications. This person should be designated the primary DCC for the department, and the department manager should be the alternate DCC. The informal organization of DCC's and ASM's is shown in Figure 4. The user first contacts the primary DCC with the problem. If he can't solve it, then the DCC contacts the IRM. The IRM has his vendor technical personnel to fall back on if he can't answer the question. Note that there is no one with the job title Alternate System Manager or Departmental Computer Coordinator. These are considered to be additional duties on top of whatever primary duties the individuals have.

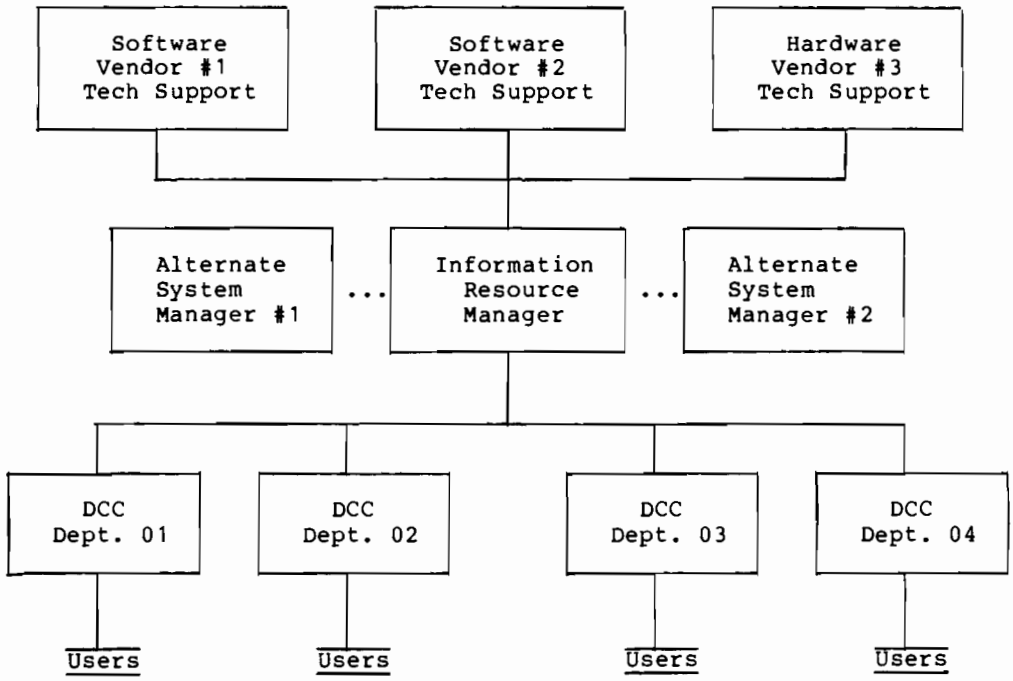


Figure 4
Lines of Communication for Problem Resolution

Finally we get to the concept of the user/programmer, which is a new word I've concocted in the style of the well-worn title programmer/analyst. A user/programmer is simply a company employee whom you've trained to use the applications software and to program with a fourth generation language. The user does not simply use; he is an active part of the total information system. The idea is to encourage user independence. Don't let your users think that the only way to get information out of the computer is to run a report written by your vendor, or a 4GL program written by another user. With just 3 or 4 hours of training on a 4GL product, most users with a basic aptitude for programming can be writing their own simple adhoc report programs. Granted, not all of your users will have that basic aptitude or even want to program. It is encumbent on the IRM to at least give all users an opportunity to learn the language and try their hand at it. You'll be surprised how many users will pick up the ball and run with it given half a chance. A word of caution is in order here. Don't let your general users have the ability to create programs that write to the databases. Designate a few key DCC's to have this ability, and let your general users use their programs to input data. Of course the vast majority of data will be input using your vendor's programs, so this caveat applies only to your homegrown databases.

You will discover after a short time that a select few of your DCC's will take a strong liking to the 4GL products. They will undubtedly be the first to approach you and ask whether the HP3000 would handle this and

that application. If the DCC shows real aptitude with the computer, then take him under your wing and show him as many tricks of the development trade as you can, within security limitations. Help him to design the database for his application, and actually let him create the database. Some of the 4GL packages, such as Dictionary-Plus from Quasar, contain utilities that allow users to build a schema in editor, then create the database and the data dictionary directly from the schema with just a couple of commands. Spend time to continue to develop these high- performance DCC's. You'll find them to be an invaluable part of the total plan as your Information System matures.

Installing the user/programmer concept at your company will render the following benefits, as a minimum:

- With users writing their own ad-hoc reports, the workload on your Programmer/Analyst is reduced, and thus his backlog will be shorter.
- Users will get a psychological boost because they feel they are an active part of the system, rather than relying on other people's programs.
- Users get exactly the report they need, since they don't have to translate their needs to someone else.
- Users get the reports they need quickly, rather than waiting the days or weeks it will take for the project to make it through the Programmer/Analyst's backlog.
- You will have depth of programming expertise throughout the company. If only one or two people are doing all the company's programming in a central IS shop, you are vulnerable in the event of their absence, temporary or otherwise.

It's obvious that if you have a hundred or so users running around gleefully writing their own programs, then you will soon have a plethora of different reports giving different people different information on the same subject. It is imperative that the IRM design some way of controlling what I call "corporate programs". These programs should produce company standard reports, and they should only be altered by IR personnel. There are many ways to design such a control system, which I will not try to outline here, but one way would be to create a separate group called COMMON that is a repository for corporate programs. The Programmer/Analyst in the IR department could be designated as group librarian, and only he could have write capability on all files in the COMMON group.

One final note on the subject of people. There is no need for a night computer operator with the small-shop concept. You can use a job scheduler like SLS or SLEEPER (which are programs in the HP3000IUG's Contributed Software Library) to run all your batch jobs overnight. The System Administrator can untangle all the output first thing the next morning and have it ready for user pickup. The only other thing a night operator might do if you had one is system backup. You can accomplish this task by doing an incremental partial SYSDUMP during the lunch hour Monday through Friday. You can pay one of your people overtime to come in during the weekend and do a weekly full SYSDUMP. Some companies do partials early in the morning, and the full backup on Friday afternoons, since most users in many companies cut out early on Fridays anyway.

RULES OF THE ROAD

The small-shop Information System based on the user/programmer concept is more than an idea; it's a reality that is working today in many companies with progressive Information System policies. Getting the system running is only half the job. You must lay out very clear Rules of the Road to guide all those concerned in order for your system to survive and prosper. The devotees of the traditional "big shop" will be swarming vultures waiting for you to make a fatal mistake if you don't follow these simple rules:

1) Do not, under any circumstances, modify any of the vendors' source code. Do not create any special programs within the domain of the applications packages.
2) Write all of your homegrown applications using 4GL products. Do not even think about using FORTRAN, COBOL, or any other third generation language.
3) Develop a solid working relationship with your vendors. They are in effect an extension of your IR department, allowing you to minimize your in-house staff.
4) Give update training to all your users, especially the ASM's and DCC's, on a regular basis, perhaps quarterly or semi-annually. Publish a user newsletter frequently between update training sessions. A well-trained, informed user is your

most precious asset. Protect your investment.

Eric W. Roberts

Biographical Sketch

Eric W. Roberts has been Information Resource Manager at The Audichron Company in Atlanta, Georgia, for the past 1 1/2 years. Audichron manufactures systems which give the familiar time, weather, and temperature announcements over the phone. He received his Bachelor of Industrial Engineering degree from the Georgia Institute of Technology in 1972. Subsequently he served in Germany for three years as a Communications Officer in the U.S. Army. He joined Audichron in 1978 as a Field Engineer, then served as Quality Control Manager before assuming his current position in 1982. Eric is currently attending night school at Georgia State University in Atlanta in pursuit of an MBA degree, with a concentration in Information Systems. He is Chairman of the SIGASK Special Interest Group of the HP3000 IUG, a group of users with interest in the MANMAN family of applications software produced by ASK Computer Systems, Inc., Los Altos, California.

VIRTUAL I.S. STAFF TO FILL I.S. STAFFING GAPS

by Ron Ellis
ARCO Transportation Company
Edited by Dave Datz

On January 1, 1983 ARCO Transportation Company's Information Services and Technology (or Systems) Department had a total authorized staff of 14 people to support four operating units having a combined total of over 450 home office employees. That staff of 14 breaks down as follows:

```
 14
- 4    Department Managers and Secretary
 10
- 3    Operations Supervisor, Day and Swing Shift Operators
  7
- 4    Equivelent Analysts/Programmers engaged in maintaining a
       variety of production systems

  3    Equivelent Analysts/Programmers available to develop new
       systems
```

On April 4, 1983 managers of the Systems Department met with the Transportation Company's Information Services Steering Committee to review prospective projects and set priorities. After thorough preparation and reasonable analysis efforts over 100 developmental projects were identified for consideration.

Clearly 100 projects is substantial, even though they had already been prioritized and evaluated on their merits before being presented. Yet, to address only the 15 or so high priority projects would require a development staff of several times the equivelent of 3 available programmer/ analysts.

A resource-vs.-demand gap such as this is probably not too unusual, and as those who have had the experience realize, maintaining creditability and providing service with a small staff can seem like an impossible challenge. This is especially true in todays environment of end users led to believe by television adds, computer salesmen and "Apple Lisa"-type computers that their needs will be increasingly met.

"OUR REMEDIES OFT IN OUR 'USERS' DO LIE"

A small twist on a line from Shakespeare (1564 - 1616) points to one of our approachs for dealing with the resource-vs.-demand gap at ARCO Transpor- tation Company. The approach is the development of a "Virtual" I.S. Staff. To provide insight into the magnitude of our efforts, a summary chart is presented in Figure RJETXT-1. It shows the number of

non-systems department personnel engaging in activities traditionally thought of as belonging to or being preformed by systems personnel. A brief elaboration on the scope and impact of our virtual I.S. Staff's activities follows.

BACKLOG REDUCTION

Our efforts to reduce systems development backlog have involved assembling and making available a variety of capabilities that end-users can use productively in developing their own applications. Currently these capabilities include:

On Our HP/3000

INFORM - A menu based, user oriented, ad-hoc report capability which is part of HP's RAPID/3000 database management package. With a minimum of instruction, INFORM allows users to select from an existing database any combination of records and fields desired and then to sort, create column totals, perform calculations, etc. as needed.

QEDIT - A powerful editing tool which is a product of Robelle Consultants QEDIT includes a word processor-like fullscreen editing feature with typical editing commands. It also has a very useful on-line foreground file execution capability referred to as "USE" files. By combining these capabilities with the HP/3000's system sorting and stream file utilities end-users with minimal training can develop and maintain some very effective list management, text manipulation and reporting tools.

SPREAD - A computerized worksheet capability that is a product of Account-A-Call Corporation and is marketed under the name Math/3000. "SPREAD" is very similar in design philosophy to the popular "Visicalc" software and basically allows an end-user to compose a worksheet with titles, headings, numeric input values, and most usefully calculations. The capability is very straightforward to learn and combined with utilities developed by ARCO allows for performing consolidations printing public- ation quality reports, and providng documentation of calculations used in a particular model.

HP/3000 UTILITIES - Capabilities such as the HP/3000's SORT utility and its facility for batch or STREAM file processing can be useful in list management applications. In particular stream files can be used to sort and select data routinely instead of using otherwise tedious interactive procedures.

On Our HP/125's

GRAPHICS - A highly end-user-oriented plotting capability that permits individuals with very little training to produce publication quality bar charts, pie charts, linear graphs and word slides. Original graphs can be produced using up to 8 pen styles and colors, and overhead transparancies can also be prepared.

VISICALC - A computerized worksheet capability that is marketed by HP. Visicalc is very similar to "SPREAD" described above. It is very straightforward to learn and Visicalc files can be interfaced with the HP/125's Graphics capability.

Use of these capabilities alone and in combination has permitted development of a wide variety of applications. These have ranged from simple list managment, spread-sheet and text processing systems that might never have been attended to by systems personnel, to entire budgeting and monthly financial or ad hoc database reporting capabilities that now do not need to appear on our already long list of systems to be developed. A summary list of our current user-developed/controlled applications is provided in Figure RJETXT-2.

END-USER TRAINING

In order to; make user-oriented capabilities available, improve the computer literary of our personnel, and provide as a byproduct what often is an important personnel development function, we have established an ongoing program of computer systems training. Designed to take prospective computer users and turn them into actual users, our offerings range from "Introduction to HP/3000" to advanced workshops where users learn to use utilities and packaged capbilities in combination. Current Information Systems training offerings are listed in Figure RJETXT-3.

The success of our training program can be seen in the statistics generated after two years of operation. Over 180 home office employees (over 40%) have voluntarily participated in a growing array of now 10 course/workshop offerings and have collectively rated their experience with an overall average 5.9 score on a scale of 1 poor to 7 excellent. However, the real test of effectivenes remains the growing list of user-developed applications such as those listed in Figure RJETXT-2.

It is of particular interest to note that seven courses or 70% of our training efforts are conducted by non-systems personnel who bring to their classes the creditability of being real users like their students.

USER DEVELOPED APPLICATIONS SUPPORT

One expected result of training new users in a variety of capabilities is their need for support. Our Systems Department lacks sufficient personnel for a dedicated staffing of user support. In fact, Systems Department staff have provided less than one half of one person in support of purely user-developed applications over the last 18 months. However, our virtual staff of I.S. instructors and I.S. Coordinators designated by major user groups, is contributing substantially to filling this gap. As might be expected, users turn first to those who trained them. Also, as developing expertise and office politics permit, users get assistance from their co-workers who know about and also use the capabilities they need help with.

SYSTEM MANAGEMENT

If users are truly the owners of their data and increasingly their production systems and applications in general, a final step in their use of systems capabilities is their participation in the management of those systems. Over last year we have formalized a concept of user account managers and conducted a formal Account Manager's Workshop. As a part of the workshop current and prospective account managers were instructed in techniques for controlling access (creating new account logons, setting passwords, etc.), controlling space (creating account groups, managing files, etc.), and supporting processing efforts (e.g. understanding user defined commands (UDC's) stream file processing capabilities, and making formal requests of operations and systems development/maintenance personnel).

Along with the above functions, specific account manager responsibilities were discussed and formalized, including.

- o First line hardware/software support and problem resolution
- o Support of and participation in emergency planning
- o Periodic file review and purge to manage system disk space
- o Periodic security revisions and checks
- o Review, comment, and implementation support for policies and procedures related to account management.

Current and prospective user account managers give our HP/3000 System Manager a virtual operations staff that combined with operations personnel, makes up an impresive group of employees able, to quickly deal with and resolve problems and to respond to new requests. Our current user account manager assignments are shown with their respective types of accounts in Figure RJETXT-4.

UNEXPECTED BENIFITS

Our success in developing a Virtual I.S. Staff to help with backlog reduction, training, applications support and system management has had an unexpected additional benifit. It has given us an increasingly computer literate user group that understands our systems development resource constraints. The result is that our systems staff is free to focus on the development of major databases and accompanying maintenance systems. At the same time end-users now expect to provide ongoing data entry and updating functions for their systems while looking forward to meeting the majority of their reporting needs with user-oriented reporting capabilities.

I.S. Staffing Gaps Filled

Creation and recognition of a virtual I.S. staff has yeilded for us what might be seen as an addition to our full-time information systems staff of roughly six to ten full-time persons. But more concretely what has occured is a growing acknowledgement of the important role our users play in meeting their own information resource requirements.

At ARCO Transportation we are hopefull that a growing Virtual I.S. Staff is just realistically positioning our actual I.S. Staff. And in this position we hope to meet the challenges of providing information services in a world where users increasingly determine and respond to their own information needs. To draw again from another of Shakespear's intuitive lines:

"DIRECT NOT HIM WHOSE WAY HIMSELF WILL CHOOSE".

SUMMARY OF NON-SYSTEMS DEPARTMENT PERSONNEL ENGAGED IN SYSTEMS ACTIVITIES

	Involvement		
	Current 12-81 to 6-83	Prospective 12-81 to 12-83	
BACKLOG REDUCTION	60*	90*	
CONDUCTING END-USER TRAINING	7	9	
END-USER TRAINING	280	400*	Participations
REPRESENTING	150	190*	Individuals
USER DEVELOPED APPLICATIONS SUPPORT	7+	9+	
SYSTEM/ACCOUNT MANAGEMENT	5	11	

*The asterisked values represent rough estimates.

Note: In NO case do the above numbers of personnel indicate more than a part time involvement with the possible exception of user developed applications.

SUMMARY OF USER-DEVELOPED/CONTROLLED APPLICATIONS

	Calcs & Worksheets SPREAD/ VISICALC	Database Access INFORM	HP/125 GRAPHICS	HP/3000 Utilities SORT STREAM	Text Editing & Report-ing QEDIT
ACCTS PAYABLE(*)		X			
ACCOUNTING PROCEDURES					X

	Calcs & Worksheets SPREAD/ VISICALC	Database Access INFORM	HP/125 GRAPHICS	HP/3000 Utilities SORT STREAM	Text Editing & Report-ing QEDIT
ADMINISTRATIVE DEPT. DETAIL BUDGET	X			X	X
AGENTS STATEMENTS (for ship"s operations)					X
ARCO MARINE INC.					
- Budget	X		X	X	X
- Monthly Financials	X		X	X	X
ATC CAPITAL ADMIN.	X			X	X
ATC MOVE SCHEDULE				X	X
AUTHORIZED APPROVERS GUIDE					X
COMMON CALENDAR				X	X
EMPLOYEE TRANSPORTATION SYSTEM(*)	X	X			X
FLEET STAFFING AND REPLACEMENT NOMINATION(*)		X			
FORMS CONTROL					X
INVENTORY(*)		X			
MATERIALS MANAGEMENT ACTION ITEMS				X	X
MEDICAL INVOICE SUMMARY	X				X
PAYROLL(*)		X			
PERSONNEL(*)		X			
PURCHASING(*)		X			
RIGHT-OF-WAY STATEMENTS	X				X
SHIPS MAINTENANCE CERTIFICATES				X	X
SYSTEM CHANGE REQUESTS				X	X
TELEPHONE DIRECTORY & EQUIPMENT ASSIGNMENT				X	X
TONNAGE BALANCES	X				X
TRAINING CLASSES & PARTICIPATION					X
TRANSPORTATION SURVEY ANALYSIS				X	X

UNITED WAY SCHEDULE		X	X
VESSEL ACCOUNTING & OPERATING STATISTICS(*)	X		

(*) Applications Involving Production system databases

INFORMATION SYSTEMS TRAINING OFFERINGS

INTRODUCTION TO HP/3000

An introduction to computers in general and the HP/3000 system in particular. Introduces basic HP capabilities including creating and copying files, and sorting. Focuses on use of the system's QEDIT (word processor-like) capability. This course is a recommended prerequisite for all other HP/3000 courses.

ADVANCED QEDIT TECHNIQUES

For users of QEDIT and system utilities (e.g. sorting, SPREAD, INFORM, line printers, etc.) who would like to learn more about using QEDIT with these capabilities. New material and areas of concern or interest to the participants will be addressed. Making this an appropriate course to repeat. Recommended prerequisite: "Intro. to HP/3000".

LIST MANAGEMENT *** NEW ***

QEDIT techniques for managing lists or tables on the HP/3000, including setting tabs, updating, sorting, searching for specific items, selecting out specific items for reports, using printers, QEDIT "USE" files, and HP/3000 "stream" files. Recommended prerequisites: "Intro to HP/3000" and "Advanced QEDIT Techniques".

INFORM

The fastest-and-easiest-to-use capability we have for developing simple reports from a database. It allows sorting, totaling, and selecting desired lines and/or columns of information. Recommended Prerequisite: "Intro to HP/3000."

SPREAD

A simple user-oriented tool (like Visicalc) which simulates a common spreadsheet and is much faster and easier than pencil and paper or even programming in BASIC. Recommended Prerequisite: "Intro to HP/3000."

ADVANCED SPREAD TECHNIQUES

For users of SPREAD who would like to learn more about how to use powerful SPREAD commands and utilities for purposes such as moving data between SPREAD files and obtaining high quality finished reports. New material and areas of concern or interest to the participants will be addressed, making this an appropriate course to repeat. Recommended Prerequisites: "Intro to HP/3000" and "SPREAD".)

HP/3000 ARENS GRAPHICS

A capability available via any HP/3000 terminal that allows you to transform numeric data and/or text into charts and graphs for overhead transparencies and report illustrations. This is a high quality graphics package that provides more flexibility than HP/125 Graphics. (General-use plotters are currently available at ARCO Center and ARCO Plaza.) Recommended Prerequisite: "Intro to HP/3000."

*** HP/125 COURSES ***

BASIC

An easy-to-learn-and-use computer programming language that provides a good opportunity to learn the fundamentals of how a computer works. Recommended Prerequisite: "Intro to HP/3000."

HP/125 GRAPHICS

A capability that allows you to transform numeric data and/or text into charts and graphs for overhead transparencies and report illustrations.

HP/125 VISICALC

A simple user-oriented capability (like SPREAD) which simulates a common

spread-sheet and is much faster and easier than pencil and paper or even programming in BASIC.

*** IBM COURSES ***

INTRO TO IBM ***NEW***

An introduction to the IBM systems used by the L. A. Data Center. Useful for any beginning IBM user. This course will be a recommended prerequisite for all other IBM courses.

EVALUATION MODEL

This model is used for evaluation of capital projects within ATC and is accessed through the Los Angeles Data Center (LADC) IBM computer. It is designed for calculation of cash flow and financial indicators. Tariffs can also be calculated under both the consent decree and original cost methodologies. Recommended prerequisite: "Intro to IBM."

If you have any comments or questions on these courses please contact Dave D 590-4500.

HP/3000 USER - ACCOUNT MANAGERS

PRODUCTION SYSTEMS	User Account Manager	Data Entry by User Department(s)	INFORM Ad-hoc Reporting
ACCOUNTS PAYABLE	N/A	N/A	X
BUDGETING & MONTHLY FINANCIALS	C	X	N/A
EMPLOYEE TRANSPORTATION		X	X
FLEET STAFFING	P	X	X
INVENTORY	P	X	X
PAYROLL	C	X	X
PERSONNEL (RESUME)	C	X	X
PURCHASING	C	X	X
VESSEL ACCOUNTING & VOYAGE OPERATING PERFORMANCE STATISTICS (VAVOPS)	P	X	X
UNDER DEVELOPMENT			
ACCOUNTING	P	X	X
COMPREHENSIVE PROCUREMENT (CPS)	P	X	X
EMPLOYEE TRANSPORTATION	N/A	X	X

USER-DEVELOPED APPLICATIONS			
FOUR CORNERS	C	X	N/A
MARINE	C	X	N/A
ATC HQ-STAFF	P	X	N/A

C= CURRENT P = PROSPECTIVE

Ronald "RON" J. Ellis Biography

As the Manager of Systems Development for ARCO Transportation Company, (A subsidary of Atlantic Richfield) Ron Ellis is responsible for meeting the information resource needs of ARCO's California based Pipeline companies and the Transportation Company's headquarters staff. Previously he was employed as the Los Angeles Branch Technical Manager for COMSHARE an international computer services firm. As a result of his employments Mr. Ellis has been responsible for planning, design and implementation of systems in over 30 companies. Additionally he has created numerous training courses to assist non-systems personnel in using computers to extend their productivity.

A graduate of the University of Southern Calif. (MBA 1978) and Calif. State University Fullerton (BA 1974), Mr Ellis has served as an instructor for Coastline Community College in Orange County. He is involved in a number of professional and community activities including the Data Processing Management Association, Toastmasters International and service as an officer of the Los Angeles Chapter of the Planning Executives Institute. His involvments in community organizations include the Boy Scouts of America, United Way, Junior Achievement and the ARCO Speakers Bureau.

The Cutting Edge: An Adventure In Fourth Generation Software

John Bescher

Limited budgets, time constraints, numerous applications to define, develope, and test. Such a scenario is faced by Data Processing Managers every day. I was not alone in my dilemma. The basic goal is to become as productive as possible and to meet the changing needs of users. I had some experience with VAX Fourth Generation Software and had found it to be very limited in scope. I was somewhat convinced that Fourth Generation Software was merely a lot of advertisements about an "RPG" type of language...sure you could put it in the hands of users, but the rest of the system would surely suffer from overloads and poor resulting response times. And certainally, the capabilities were severely limited. At best such tools could only be used in conjunction with "normal" programming methodology: Cobol, Basic, etc. However, because of the large quantity of applications development backlog and the fact that I could not hire hordes of programmers, and because most of the applications were custom, I elected to invest in Fourth Generation Software tools. I wish to share my experiences with you on my successes and failures with Fourth Generation Software.

Before discussing my adventures with Fourth Generation Software I would like to briefly describe the environment. The applications and the users at the Signal Companies are rather unique in some ways and common in other aspects. The environment is identical to a small 150 employee company. There are needs for a Payroll, General Ledger, Accounts Payable application for the activities of these 150 employees. On the other hand, the facility is the corporate head- quarters of a 7 billion dollar, over 73,000 employees, corporation. Many employees are Very Important People who are independently wealthy and where salaries exceed $500,000 per year without counting bonuses. The General Ledger System must contain up to 9 billion dollars per line item or journal entry. The typical user is not engineering, nor technical, nor manufacturing oriented. In fact, most users are first time data processing users.

The Signal Companies, Inc is a multi-industry company with strong, strategic positions in the aerospace, electronic communications, energy service, transportation and construction industries. Signal provides sophisticated technology and engineering services and high-quality products to these and related industries through- out the world. Signal is 60 years old and has its roots in the oil and gas industry. Signal was formally known as "Signal Oil and Gas Company"

Signal is an exceptionally strong, high-technology company with powerful earnings capacity. It ranks between 45th and 50th on the 1982 Fortune directory of the largest U.S. industrial corporations. Signal's sales approximate $7 billion, its assets total $5.4 billion, with over $2 billion access cash and lines of credit. The corporate headquarters is located at La Jolla, Calif, a suburb of San Diego. I installed Fourth Generation Software at this headquarters.

With the construction of this headquarters in 1980 the Chairman of the Board and Chief Executive Officer, Forrest N. Shumway, created an environment that would provide comforatable working conditions and maximum efficiencies for the approximately 150 employees at the facility. These employees consists of very high ranking executives, attorneys, public relations specialists, and staff. Banking activies, tax concerns, stock acquisitions, divestitures, legal corporate decisions, mergers, corporate finance, insurance, overall company policies, investments, and employees savings plans are all orchastrated from this headquarters.

The Spanish-California style architecture was selected in conformity with the San Diego/La Jolla area. Combining location, which is on the Torrey Pines ridge overlooking North San Diego county, architecture, interior design, and landscaping, the result is one of the most outstanding and impressive corporate headquaters in the western states.

In addition to many private offices, a large dramatic lobby, this 100,000 square foot building also has an employees' dining room with a fully equipped restaurant-type kitchen. The forty foot long board of directors room is impressive with its massive arched windows. The facilities include a wine cellar, photolab, executive dining room, medical wing with a full time physician, nurse, and x-ray equipment. Employees can also use the recreational facilities during non-working hours: a heated pool, spa, 2 tennis courts, racketball court, and exercise room with body building equipment.

This environment emits a feeling of quiet power and wealth. Being from a classic data processing environment, I was somewhat concerned about fitting in...will I spill coffee on the customized Persian rugs? How will these VIP's relate to "computer" technology? Will computer equipment and operations be delegated to the basement out of sight of the high ranking executives and their priceless antiques and paintings? On the other hand, I was intrigued on how computer technology could improve operations and efficiencies with- in the headquarters.

I was hired early in 1981 to create a data processing capability within the headquarters. Certainally there was some data processing operations already available: batch processing on a timesharing system. Computer capabilities were limited to card oriented batch input via a Remote Job Entry station with resulting printouts. Most of the applications were manually manipulated: A Payroll system, General Ledger, and Accounts Payable on the remote timesharing system with a mixture of customized Cobol applications. The Hewlett Packard System 3000 had arrived in a crate during my interview. It was sitting forelornly at the end of the first floor hall when I arrived for my first day at Signal. Smith Dennis and Gaylord, a software vendor in Santa Clara, was contracted to provide most of the financial software: Employee Savings Plan, Executive Stock Awards, General Ledger, Payroll, Accounts Receivable, Accounts Payable, and Fixed Assets.

I drafted a five year plan and distributed it to the staff outlining the directions for the new Hewlett Packard System 3000. It consisted of three phases: First the installation of the financial packages from Smith Dennis and Gaylord, next the procurement of Fourth Generation software tools to create the "other" applications that were either weakly defined in the prior discussions with the software house or not defined at all, and Third to create a link with subsidiaries via telecommunications. At this point the plan has been beat...the telecommunications system is underway using Remote Job Entry, IBM personal computers, a bank of modems, switches, modem eliminators, and a centralized tandom pair of IBM 3033's at Signal's Kellogg Computer Services in Houston, Texas.

Against this background, I now would like to share with you my experiences with Fourth Generation Software. The envirnoment is a very powerful corporate headquarters with a limited staff. Users are non data processing oriented. Probably typical. They had suspicions that Data Processing people were high technology "egg heads". Data Processing was very expensive, reports were never quite right, any development request would be met with blank stares or impossibly long lead times. Prior to the advent of the Hewlett Packard System 3000 the Cobol programs customized on the timesharing system were sometimes over 6 years old...and of course the programmers were long gone. Operations were based on folklore, heresay, guesswork, reruns, and card decks used over and over again with little understanding of what was actually happening. Timesharing rates were increasing with little control by the user. All data and input were the responsibilities of the Data Processing staff. Lastly, even the easiest appearing change or request for a new report was looked upon by the Data Processing staff with horror. Within the first few weeks of my job, I was asked by a senior vice president for a report showing tax data sorted a different way then the usual report. I thought the task was reasonable, but to my embarrassment found that the key needed to sort the data was not included in the data base, nor would ever be in the data base without a lot of conversions, program design and recompilations, and tests. I found the report would take several weeks to develope using the existing serial file structure on the timesharing system and Cobol. The VIP, of course, needed the report for a meeting the next day. Without going to extraordinary efforts (emergency consultants, an all night effort, etc) the report would have to wait. I vowed to remedy the problem with a Fourth Generation Software tool on the Hewlett Packard System 3000. Eventually all applications would be removed from the timesharing system...thusly, all effort except emergencies would be put on the System 3000.

In selecting a Fourth Generation tool I examined several packages. Utimately I selected Quasar's Powerhouse because I felt I needed ease of use and friendliness above all....even above computer efficiency, response time, etc. I was in a hurry to impliment some applications quickly with as few resources as

possible. In retrospec, I believe any of the major packages could have been used successfully at Signal. I still think, and this is just my opinion, that the Powerhouse package is the most friendly, complete, and practical for my environment. Lastly I wanted a package that I could safely share with some of my users. I wanted some of the non-data processing employees to have report generating capabilities and perhaps, even creation of applications. My ultimate goal was to leave the data base definition, initialization, and maintenance to Data Processing and let more sophisticated users generate screens and reports for the data base. The vendor did not believe this would be a good idea. They felt users were capable of inocently destroying data, errorinously using Powerhouse, and in general, causing an uncontrolled series of data processing catastrophies. I began to feel the mistrust was equal on both sides: users thought programmers were arrogant, overpaid, and incapable of a sense of urgency on even the simplest request. "Computer" types thought users were scatterbrained, disorangized, unsophisticated neophytes incapable of understanding the complexities of data processing.

In a sense, my comments are not only directed towards the adventures of Fourth Generation Software, but also in reducing this attitude friction between users and Data Processing staff.

For those who have not been exposed to Powerhouse it consists of:

-A dictionary
-QUIZ: A report writer
-QUICK: A screen generator used to create menus and screens to create, maintain, and delete data base records.
-QTP: A program generate that can be used to make sweeping changes in a data base, editing an entire file, or producing logical updates, additions, or deletions in a data base.
-A series of utilities to compile, list, and integrate dictionaries, screens, and programs.

Powerhouse is capable of defining, creating, and maintaining data bases for MPE, KSAM , and/or Image files.

Both ample documentation and a call-in consulting service are available.

The Lure and the Promises of Fourth Generation Software!

I had expected Fourth Generation Software to reduce definition, creation and testing time for customized applications. I also expected a resulting application that would be homogenous to all users, easy to use, and a breeze to maintain. By homogenous I mean the same key functions, help message operations, format of screens, exit techniques, etc. between all applications. A user would not have to be retrained when going from one application to another. I would not have to remember, for example, that "exit" ended one screen, "End" exited another, the "F1" key another, and upper case "6" another.

In the almost two years of using a Fourth Generation package I found that the biggest gain was creation of "ad hoc" applications. Often, I would receive a vague description of a desired application from a senior VIP and turn over the completed application to his secretary in just a few hours! The speed of developing such "ad hoc" applications is very gratifying to both programmers and the user.

On the other hand, there is a big need to improve definition methodology. I do not know which methodology is best. Interviewing users is the first step in understanding an application. Unfortunately there is the innocent unsophisticated user who knows the application but who forgets to tell the developer all the vital details which if not included in the data base makes the resulting application worthless. The other type..on the other end of the range...is more sophisticated and wants to know why the masters are such, and why detail files, and why not put these 200 fields on one record, etc. They volunteer too much and thusly bog down development time. Endless discussions develope with these users about whether to put parameters in a control file or embed them in screens. This user seems to always have new ways of producing "better" application updates, reports, logic, etc. Inevitability one must either ignor such a user or give in to a particularly outlandish idea just to prove to the user it was a bad decision. Either path is a

poor choice. And sometimes this type of user is correct...which feeds his appetite for more information on how the system works. Unfortunately this user sometimes neglects his real job for the intrigue of data processing...then in the midst of an application that he has masterminded with his own techniques he has to return to his primary job. At that point the project is orphaned.

Definition, then, is a still major task; and if not successfully completed, will be a fatal flaw in the project. The major benefit of Fourth Generation Software for definition phase is the ease of creation...and resultant review by user. I hope the universities help us in providing better ways of defining an application because the state of art right now still leaves a lot to desire. The gap between user and developer is very wide here. Not only are there misunderstandings, wrong assumptions, forgotten conversations, and changing requirements; there is always the very real possiblitily of the user not really knowing the details of the application. It seems humans are not very good at details. Computers need details.

A Commitment in a Corporate Environment

When I purchased the Fourth Generation Software package I made the commitment to myself that I would use the package for all new applications on the System 3000 unless an application was already available from a software vendor. I had been told by my friendly salesman that the package could do anything. This was a challenge to me...I had to find an application that Fourth Generation Software (at least this product) could not solve. This excluded communications software such as RJE.

In meeting this commitment to use Fourth Generation Software for all new applications I had to resolve several issues:

-Is Fourth Generation Software really practical for all applications? In other words, are there some applications that are too complicated or require logical operations not available with Fourth Generation Software?

-In spite of the fact that Fourth Generation Software requires less resources per application, I still need resources (people) who know both the Fourth Generation Software and the application requirements. Where will I get those resources?

-In charting this path, will I end up with an overload to the system? Is there a way I can measure the load on the system and thusly change my goal before it is too late?

The results? The commitment was kept. I found many applications that initially were too complicated for Fourth Generation Software. I found several old forms and reports that could not be identically duplicated with Fourth Generation Software. I experimented a lot. I called the software vendor often. When stuck on a particularly sticky problem I retained a consultant from Los Angeles for a week. In the end, I found solutions for all applications through Fourth Generation Software. I admit, sometimes I had to spend extra time with the users explaining why I could not produce an identical report, but we always found a compromise. And sometimes the compromise report was a better report than the original.

The solutions to resources and performance fears are many and direct: In order for Fourth Generation Software to work it must have a driving force: an individual who is determined to use it and will encourage others to use it. I became the champion for Fourth Generation Software. After attending vendor's classes and using it myself for several applications, I taught local users including my programmers and operators. I set up example source statements to use as standard screens for others. My biggest disappointment became the lack of local consultants. I could retain Cobol, Basic, Data Base experts...but in San Diego I could not locate any consultants for short term work who were knowledgable with my particular package which, by the way, is the most popular Fourth Generation Software package available for the HP 3000. Through the assistance of the vendor, I was able to locate several experts in Los Angles. I hope with the growth of data processing in San Diego the availability of such consultants increases.

As far as performance fears go, I could not find a fool-proof way of determining how badly I was loading down the HP 3000. Sure, I could measure percentage of time spent on I/O and CPU utilization. But, response time is a very elusive concept. I still have no way of knowing that I will, for example, increase average response time by 100 percent next month based on the past year's increases. In fact I do not even know my past year's increases. At this point I have several jobs (Basic programs written by consultants before I purchased the Fourth Generation Software package) that run for hours and hours. However, in general, I have not had complaints from any users on interactive response time delays! Will the very next application installed cause unacceptable response times for every user? I honestly do not know. All I can state, is that in spite of over 12 major applications written in Fourth Generation Software, none have ever been unacceptably slow or caused deteriating delays for other users.

To date the following applications have been created at Signal headquarters using Fourth Generation Software:

-Medical Insurance Claims
-Subsidiaries' Insurance Contributions vs Claims Cost
-Self Paid Insurance Cost History
-Public Relations Multi-Industry Mailing List
-Cash Management
-Accounts Payable Vendors
-Debt/Credit Data Base
-Dental Claims
-Income by Industry
-Bank Agreements
-Savings Plan Auxilliary
-General Ledger Auxilliary
-Payroll Auxilliary
-Stock Analysis

Planned applications include:

-Bank Accounts Directory
-Employee Expenses AP-AR Loop
-Company Car Maintenance and Cost
-Political Action Committee Eligibles
-Docket Scheduler
-Foreign Exchange Exposures Interface

The Cobol Trap. Conversions.

I had a choice. I could convert existing Cobol programs running on our timesharing system to Cobol for HP 3000 or I could use Fourth Generation Software tools to rewrite the applications. The programs had to run on the local HP 3000. The users were unknowledgeable and complacent...in the past they had merely filled out a form or called our Data Processing operations to run a job and, in return, had eventually received a resulting report. They could care less where the software resided. I was attracted to the apparent ease I could take these Cobol programs from IBM to HP. With very little effort I should be able to run these applications on the in-house HP 3000 by Cobol conversions...after all, one of the advantages of Cobol is its portability. I decided to try the "easy" conversion and then a Fourth Generation rewrite of just one of the applications; measure the results in terms of a product and time investment; and finally, decide which direction for the remaining 6 to 8 applications. Fortunately, I had joined the Hewlett Packard Users Group and learned about an opportunity at San Diego State University: For senior Data Processing students the university offered a course of field work. Students would get a chance to work in a live data center environment and earn credits. I could have students working on my experiments without any cost to the corporation! I assigned one group of students to the Cobol conversion. The next group to the rewrite in Fourth Generation Software. Two major applications were selected: a complex public relations mailing list of over 7,000 entries and a legal court docketing date scheduler. The result...the Cobol conversion too longer than anticipated and the final version on the HP 3000 operated just like the original timesharing version: batch with many restrictions: a poor product. Maintenance was going to be a problem. It took two students working 16 hours a week two months for both conversions.

On the other hand, the rewrite using Fourth Generation Software took the same time but resulted in a much more flexible mailing list application. During the rewrite of the court docket scheduler a major change occurred within Signal which made the basic applications applicability questionable. Essentially, it was better in the long run to rewrite using Fourth Generation Software than to attempt the conversion. With the conversions the users seemed to care less, with the rewrite they were pleased with the enhanced product.

My commitment to write all new applications with Fourth Generation Software was underscored! It works! I almost fell into the trap of the "easy" Cobol conversion...only to find it was not that easy and that the resulting software would (surprise) be no better than the batch 80 character input, non-inquiry, poor maintainable, timesharing version.

Plunging in where Angels Fear to Tread

Even the enthusiastic vendor salesmen was negative on letting non data processing individuals use the menu and screen generation Fourth Generation Software utilities. With these utilities a careless user could easy purge, destroy, change, etc data base information. The vendor guideline seemed to be "let the user have non-destructive report generation capability only". Only Data Processing personnel should have the capabiltity to create software that could change data. I presume this attitute was either to protect Data Processing personnel from being blamed for a destroyed file when the user did it unknowingly himself...or its the old job security syndrone. From the beginning I had envisioned violating this guideline if I could find some users who would have both the patience to learn about Fourth Generation Software and the common sense to keep himself from doing foolish things. For those individuals who were trusted with tools capable of changing data I made some disclaimers:

- I would be available as a technical adviser...but if questions became too frequent, I would take over the application.

- Only Data Processing personnel would be permitted to call the vendor for a resolution to a problem. I wanted all such calls to go through a senior analyst both to avoid unnecessary calls to the vendor and that any problems and resolutions would be shared...the next time that same problems occurred with a different user a senior analyst would already have the answer.

- Outside of guaranteeing the integrity of the hardware, backup operations, and Operating System soundness I would not be responsible for the data manipulated by the user created applications.

Users now having the capability and knowledge to create their own menus and screens include all data processing operational personnel, the Manager of Finance, the Assistant Tresurer of the corporation, and the Manager of Accounting. Thus far, no problems have been created via this arrangement. In fact, many of the former problems have vanished. If my resources are tied up in another application, some of these users can start their own screens and menus. This gives me some breathing room to finish one application and start the new one. Also, many users now appreciate the need to define applications before they are programmed. They have a better understanding of what it takes to develope an application.

I wanted to spread Data Processing appreciation to the entire staff. Because of the many manual operations within the facility I felt there were ample applications that could be computerized with Fourth Generation Software. The biggest problem I have, however, is defining those applications in sufficient detail to start the software creation

phase. By holding seminars on Data Processing concepts I thought I could shorten the definition time requirement by educating users on terminology: fields, records, data bases, etc. Even such elementry concepts such as sort keys, numeric versus character, master records versus detail records, etc were foreign to these users. If users would understand these elementry concepts it would be easier for me to learn about their applications. We seemed to have two worlds: data processing terms foreign to users and application terms such as Coordination Benefits, Cut-out, etc that were foreign to me. The series of seminars; unfortunately, were unsuccessful. Most users did not have the time, were confused, promply forgot, could care less, or, in a few cases, had to learn everything about Data Processing for it to make sense to them. For those people, an incomplete understanding of all aspects of Data Processing seemed to make them apprehensive and uncomfortable.

In the future I will change my focal point for these in-house seminars. Rather than try to acquaint users with elementry data processing terms and concepts I will try to show them capabilities and possibilities of the system. Not how..but what. Not about signed numbers, master and detail data sets, why it is important to be able to identify each employee record by an identification number which is unique instead of the person's name...but rather what kinds of applications have already been created on the HP 3000 and the drugery and paper work eliminated by those applications.

I realize that changing the focal point of the in-house seminars will only increase the number of applications destined for the HP 3000 and will not help in the definition of those applications...but as I said before, I need a better methodology for defining applications...and seminars do not help significantly.

The Performance Poltergeist.

In the beginning I had opted for very friendly Fourth Generation Software over efficiency. There is a range of efficient software starting with assembly language and SPL through Cobol, Basic, Pascal and ending with Fourth Generation Software. Even with Fourth Generation Software some packages are more efficient than others. Generally, it seems the more friendly the less efficient the resultant software: efficient in terms of response time, load on the system, run time, etc. I had elected for friendliness in the extreme. I did not want to spend a lot of time learning about the new package...I want to create applications. I was nervously wondering if, in the long run, I would have to pay the price. So far, the threat of poor performance has just been a "poltergeist"...it has not materialized. One saving grace is that most of these applications in Fourth Generation languages are interactive, most have small data bases, and most are only weekly or monthly activies. I have not had a single complaint about response time or processing delays. Most of the terminals in the facility are still set to 2400 baud. In one case the user wanted to slow down the terminal because his reading speed was outmatched by the application.

To protect the system from future loads I have ordered disc caching for installation sometime in 1984. Because I have a basic model 40 (the least powerful processor) I can always upgrade the processor. Lastly, I can order more memory (I have 786K now) or another 404mb disc (I have 2 404K and 1 120 MB). With that room for growth I feel protected from any performance poltergeists.

A Stranger in a Strange Land:
Interfacing Foreign Applications

One of the most productive and rewarding uses of Fourth Generation Software at Signal headquarters has been foreign software interfaces. There are several applications written by a software house in Basic using an Image data base. Maintenance of these applications is done by the software house. In many instances I have been requested by users for short reports not available from the existing software. There also have been requests for changes to the data base based on logical constructs that were not in the existing package. Without Fourth Generation Software I would have had to request the software house to provide those reports and programs...at considerable expense and time delays. With our Fourth Generation Software I could "share" the data base with the software house package and provide reports and generic changes not possible within the time constraints imposed by the software house. The purchase of the Fourth Generation package could have been justified just in this

interface capability! I have used the report generator to test the software house's new releases by comparing our report against the software house's report. I have saved the corporation over $20,000 in "non-standard" reports....reports that were not defined when the initial application was given to the vendor for development. A new project is underway to provide an Accounts Payable link to Accounts Receivable using Fourth Generation Software.. such a link would have cost over $12,000 from the software house in its current undefined state. I am sure that cost would have been tripled after the definition is firm.

In many instances the use of Fourth Generation Software in conjunction with an existing application makes not only sense, but becomes mandatory for important, fast changing application environments.

User Involvement: How much is too much?

How much information does one give a non-data processing individual? How much capabiltiy? At what level do you keep data processing secrets? At Signal I felt that the HP 3000 system was common property...I did not own it, nor did anyone else. It is a facilty to be used by any appropriate employee. Data Processing was responsible for the integrity of the system, its operation, backup, current operating system, etc. Data Processing is responsible to ensure the correct tools, utilities, and methodogy is available and used. Security of the system is also a Data Processing responsibility. The content of the disc files and data bases, however, is the responsibility of users. I did give several non-data processing employees the capability of managing their own data...they could create their own menus, own screens, and own data bases in addition to their own reports. I have not had a single regret for giving these users such powers. On the other hand, only Data Processing personnel are assigned "manager" user-id's and password knowledge. Some of the adverse effects I have experienced from this arrangement are:

- I get a lot of telephone calls on "how to" questions. Most of the time I have the answers. For those times I did not, I found the resolutions and learned more about Fourth Generation Software. Such calls can be frequent, disruptive, and occasionally annoying. In the long run, however, more application work is being done then if I kept the work to Data Processing personnel only.

- Infrequently I become irritated that the software is not more user friendly. In spite of the quality of Fourth Generation Software there are many detailed technical implications...beyond what a normal user needs to know. For instance, "RTIO" must be specified in MPE files in order for deletions to work. Even Sometimes odd boundaries (versus even) causes system confusion for Image files. In the future, these "gotch ya" pitfalls will be eliminated from Hewlett Packard resident Fourth Generation Software making it totally tolerant for non-data processing people.

- Just like computer games and Person Computers, many users are intrigued by Fourth Generation Software. They want to know how, why, when, etc. When they learn they become "experts" and like real experts they do not take advise any longer...and run into unexpected (but predictable for the experienced) problems. Meetings with such users can be very lenghy. Instead of understanding the application being defined the meetings can disintegrate into a discussion and debate on computer methodogy...should this be a master, detail, auto, etc data set, how many passes will we need to generate the report, how many programs do we need? Sometimes the Data Processing staff can take offense at this intrusion

into their domain. Often meetings become unproductive. Users begin to spend more time on the programming tasks than on their real job.

There probably is no real guide on how involved a user should be. In some cases, the user can be completely involved...let him write his own application. In other cases, especially with very complicated, far reaching, and large data base projects involving several departments, the design, development, and initial test should be solely the responsibility of Data Processing with the other departments assisting in the definition of the application.

Emergency Applications: Heros on White Horses!

There is nothing like being asked for an emergency software request and being able to deliver in an unbelievable short time. You become a hero on a white horse, a wizard. Somehow with some magic you did the impossible. With Fourth Generation Software such magic is possible. One afternoon I was asked to deliver to a very high ranking executive a summary of several interesting companies showing outstanding number of shares, price of stock, assets, etc. Such information which is publicly available would be used for Signal investments decisions. In two hours I was able to define and create a data base, construct menus, build entry and maintenance screens, and reports. The system was then turned over to his secretary for data input. Such a feat would be impossible in the amount of time for Basic or Cobol. Needless to say, the executive was impressed.

There are times when an application report seems to be erroneous or the information is available on disc, but no existing report shows the summary information just the way it is required at that time. At that point a Fourth Generation report can be quickly defined and executed giving the anxious user the information needed to find out why his report is out of balance, showing no entries, or faulty entries and summaries. Given the time spent on installing and testing the initial Basic programs, users are often astounded at the rapidity of a new report being created by Fourth Generation Software. My most rewarding and exciting efforts have been for emergency situations using the Fourth Generation Software tools.

Prototyping. Specificiations: The Jellyfish Application

As I mentioned previously, the biggest problem I have is in the initial definition stages of a project. I tried a series of seminars so users could understand Data Processing concepts. That ended with what I felt were poor results. There was just too much to learn in too short a time. Most of the applications were started with several meetings..almost interviews..with users so I could understand the requirements. A prototype would be built and the users would review the structure. Changes would be made and the cycle would be repeated until, hopefully, the "correct" solution was available to the user. Problems I encountered are:

- Sometimes the prototype-review cycle would never end. Changes would be heaped on changes resulting in a very "patched" or "dirty" file structure. Eventually, the file structure would have to be redefined.

- Major changes often were reveiled by the user which rendered the original design obsolete. Sometimes these changes were simple omisions, but frequently they came from the increased awareness by the user of the capabilities of the system. "Wow! I didn't know you could calculate those summaries...That's exactly what

I need. Now, if you could only link with those summaries the subsidiaries' contributions." Redesign, compromise or freeze the design! Because some users plead so well for a new found capability, often its redesign.

-In frustration, I have had to freeze the definition-prototype-view cycle for at least one major project because of the mounting changes that were occuring over several months. I regressed back to written formal specifications which were distributed to the staff for sign off. This has happened twice for this particular complex project. The user just could not make up his mind...or things changed. This project is now in hold pending formal signoff from the user. This emerging and changing design causes a "Jellyfish" project: half defined, half prototyped, ever changing, needed but obscure.

-Another potential problem arises when the user becomes too excited about the new application and begins to use it before its properly tested. Two frustrations are common with premature applications: 1) other departments get reports that are erroneous and therefore embarrassing (one user was so thrilled at the premature..and faulty...report that she hand corrected the incorrect totals on the report and sent them to other departments!) and 2) the user spends hours upon hours inputing all the data for the data base only to discover there are major changes to the data base. In that case, a conversion is needed instead of just re-entering a few test records.

-Always leave room in every record for expansion when prototyping...I define at least 20 characters per record for unknown, future fields. That way the data base does not have to be recreated for testing. One merely has to redefine the unused field for added status flags, company codes, totals, etc that were unfortunately overlooked in the initial design review.

-There is an axium in data processing that one must always throw the first design away. With Fourth Generation Software it is easier to throw the first design away and start again because the data base creation, menu, screen, and report generations are usually very easy to accomplish. Less is invested in system creation efforts, therefore the pain to discard the first version is bearable.

Junior Programmers let Loose

How much capability does a Data Processing Manager give to junior programmers? A little bit of knowledge can be dangerous. I have found most junior programmers to be very eager...full of energy. They are ready to conquer the world. They are confident they can do anything! And there lies the trap. Without realizing it, they get over their heads with complexity (as we all do occasionally) but rather than ask for help...or even recognize they are in trouble...they continue to storm ahead. I have had several student programmers work with Signal as a result of the San Diego State Universtiy student work program. In general, their accomplish- ments to Signal applications have been significant and I hope they have gotten a taste of the real data processing world by working at Signal. My observations are:

- We in data processing seem to be generally plagued with underestimating the effort to create applications. This becomes worse with junior programmers. In spite of Fourth Generation Software there remains many "gotch ya's" in data processing. Obscure problems that take time to resolve. For the students on loan to me from SDSU the underestimation was not only worse, but the students would return to school after a set time...whether the project was complete or not. It was up to the Data Processing staff to pick up the pieces. If such students are used, one must assume that the staff will take up where the students left off.

- The students (junior programmers) were assigned maintenance projects or were given models to use for new applications. This worked out very nicely, especially when software documentation from the vendor was comprehensive and descriptive.

- In the few cases where I let more aggressive students try to work with users in defining an application..and then work with me in laying out the data base, I found to be marginally successful. Users wanted to talk to more experienced data processing personnel. For some reason I felt a slight resentment by the users for the junior programmers. I could only attribute such resentment...and it was slight... to the fact that the junior programmers were just starting a career (Data Processing) that was very envious to secretaries, word processing clerks, and clerical staff members.

Conclusion: Promises Kept

By describing some of my adventures with Fourth Generation Software I hope you have detected my enthusiasm for such software in the type of environment at Signal: Standard vendor supplied financial packages, non-data processing oriented users, a small Data Processing department, with many customized applications requested by various legal, public relations, banking, tax, and financial staff members. In spite of threats of system overloads and applications complexity that could not be resolved by Fourth Generation Software I have found pleased users and more timely application development. I have cautiously allowed non-data processing personnel create their own menus and screens...with no regrets. I have let junior programmers maintain Fourth Generation Software applications with little supervision...with no regrets. And I have avoided, with one exception, the time consuming, dull, ritual of specification writing, review meetings, and reissues of specifications, sign-off, etc by wide use of prototyping...with no regrets. There are still a lot of pitfalls with Fourth Generation Software: technical aspects that need not be burdened by the user such as RTIO, Stack Size, Word Boundaries, and frequent need for double pass operations. However, we have come a long way from the laborious Cobol programs and reliance on programmers to create, maintain, and change user's own data.

In closing, I would encourage all of you who do not have Fourth Generation Software to explore the practicality of such a tool at your facility. You are doing yourself and your company an injustice if you do not at least investigate this proven way to make your staff more effective and efficient.

Electronic File Cabinet -

Online Information Systems for unstructured Data

by Franz-Josef Boll and Joachim Geffken

Introduction

Most of today's computers store large amounts of text. These are typically unstructured data resulting out of day to day computer usage in word processing, documentation, office automation etc. Whereas data in structured files can easily be accessed by Query type software, documents, like unstructured data in general, require special retrieval techniques. This paper will present the "biography" of a document retrieval system - the package IDT - EF (Electronic File Cabinet). And it will discuss some of the general principles of document storage and retrieval, like searching, sorting, user interfaces and aspects of implementation.

IDT-EF has been developed by Franz-Josef Boll at Herbert Seitz KG of Germany in close cooperation with Radio Television Luxembourg (RTL). Although it is a general purpose software, a short overview over RTL will help to understand the scope of the software.

Radio Luxembourg - A Large Media Company

Radio Luxemburg (Radio Tele Luxembourg) is a large private owned comercial media company in Europe. They broadcast radio and television programs on many channels for different European Languages. They are the largest private owned company of that type in Europe. RTL is also engaged in cable tv and most recently in satelite tv.

Large amounts of data, mostly unstructured, have to be handled in the administration of texts, photos, audio and video tapes and films.

Creating a Retrieval System

One of the major challenges was the film archive: a large number of short films transmitted every day as part of the newcasts. Computer usage was necessary to handle the archive efficiently.

We met that needs by creating a document retrieval system, where the documents contain all the keywords required, secondary information and references to the audio and video information stored in different archives.

When we started working on this, we already had the experience of creating a large word processing system. From that, we had a large number of subroutines that we could use.

Very helpful to us was RTL's experience in this field. They had done extensive investigation on retrieval systems. They provided us with many ideas about the retrieval system. Thus it was possible to set up the system in a few months instead of a couple of years. **IDT-AS** has in the meantime proven that it meets the requirements of RTL as well as the requirement on a general purpose retrieval system.

Descriptors and Selectors or How to Retrieve a Document

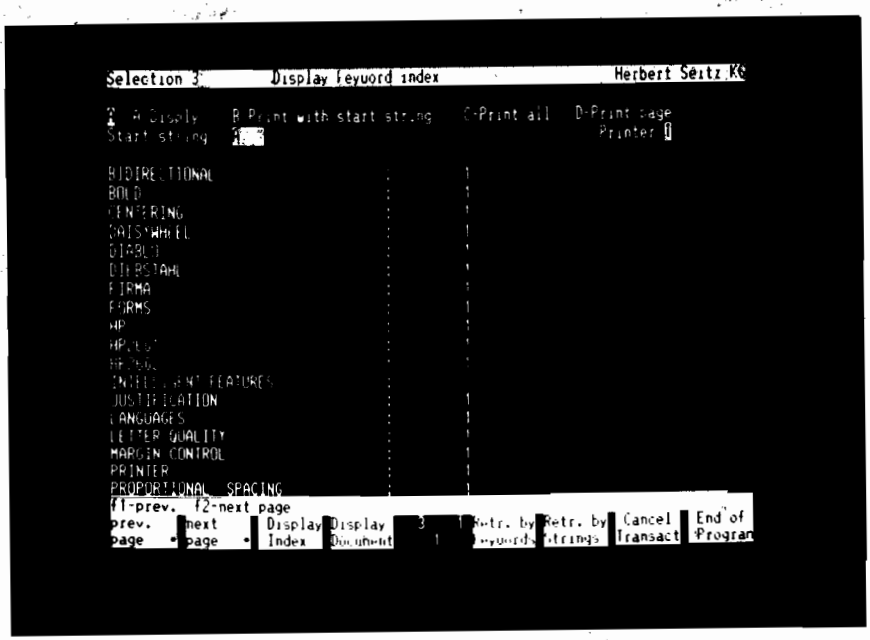
The more documents you want to store, the more useful is a document retrieval system. Large systems are only manageable with computer support. As document data bases gain substantial sizes, sophisticated retrieval algorithms are required. Even in extremely large text data bases retrieval should be fast - and accurate. That means with todays computers (RTL uses hp 3000-44's) you have to be able to retrieve a small set of documents out of 10,000 or even more. All retrieval should be done online. Response time must be in the range of a few seconds for standard and a few minutes for infrequently used transactions. There are basically three categories of information that identify a document:

- Keywords
- the sequence of characters forming the text
- and secondary criteria, like the document's name.

The efficient usage of these three retrieval types and an appropriate combination of them should yield not only a fast response, but also an accurate and precise information.

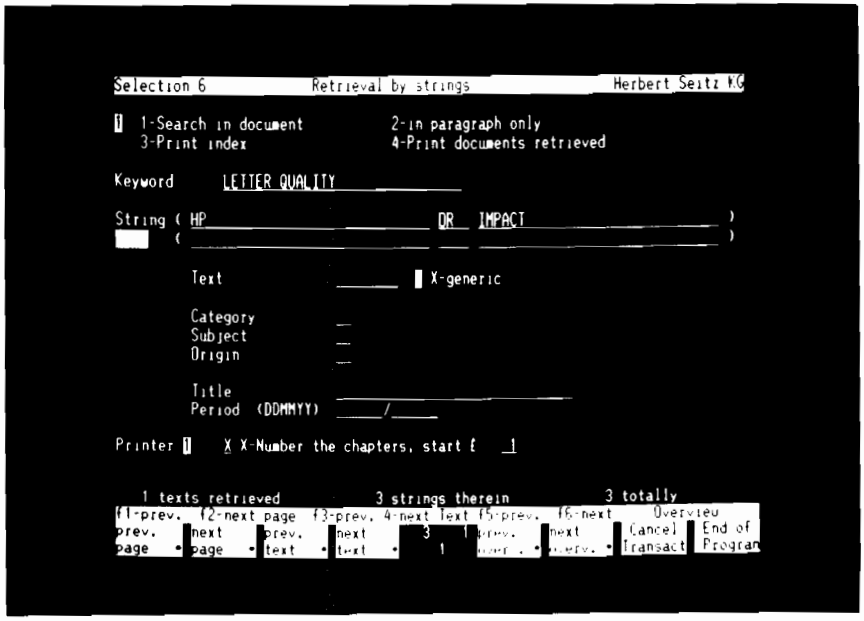
In the following paragraph we will discuss these three categories in more detail:

- Keywords:
  This means that you maintain an index - a thesaurus - of descriptors related to individual documents or parts of it. This approach gives the best response time, however, it requires the storage and maintenance of a keyword index. Anyhow, this approach is a must for extremely large text data bases - we will show later how index maintanance can be done at least semi-automatically.



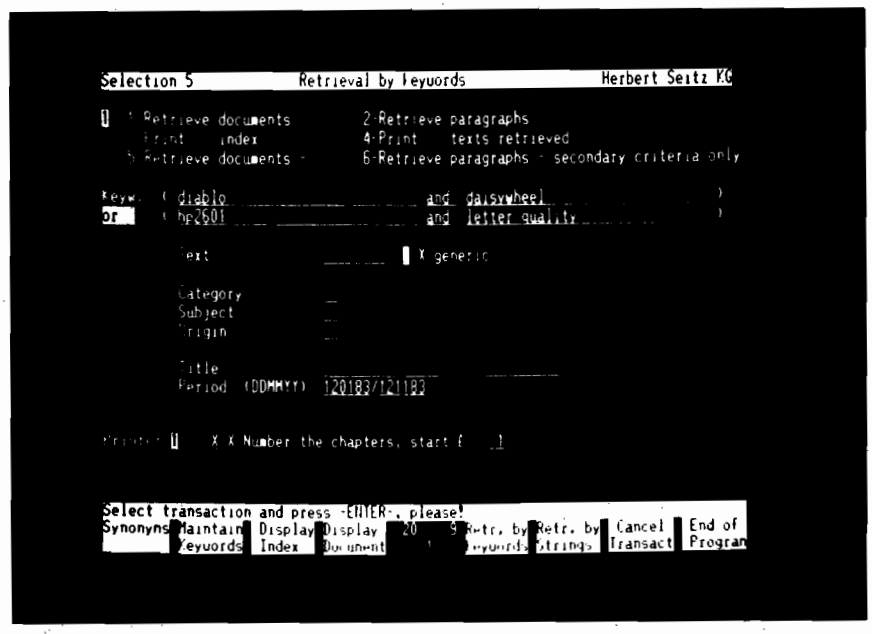
- Character string patterns:
  This approach is very flexible, as no predefined index is used. Every information within the text, every character string serves as information and thus as possible retrieval element.

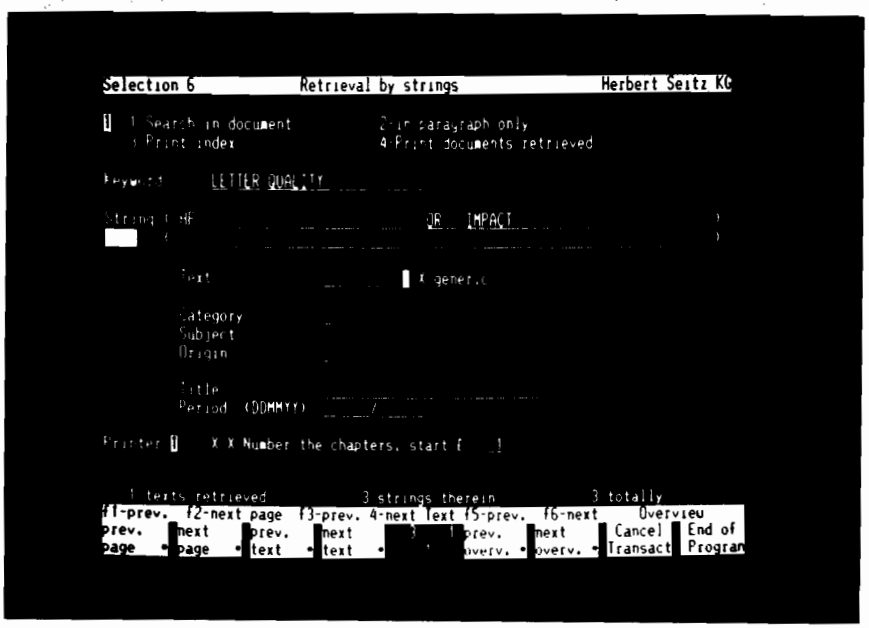
  The associated problem is speed. The IDT-EF currently is able to scan about 1000 to 3000 lines per minute (on a hp 3000-44). This includes reading text, upshifting, concatenating hyphenated words, searching and selecting.



- Secondary criteria:
  Every document, once it is created, has some criteria related to it such as the document name, creator, title, theme, date of creation. The IDT-EF maintains a number of additional secondary criteria such as titles, categories and so on. Secondary criteria usually are not random access criteria (except the document name), but they are a means of making retrieval more concise and faster, as they divide a large number of documents into a subset that can be substantially smaller. Especially we would like to mention a feature that we call "generic document names", a fast way of retrieving meaningful subsets.

With the IDT-EF you can use each of these retrieval methods separately, and you can combine the three methods at the same time, thus allowing a maximum of flexibility, accurateness and speed.





Extended Search Methods

The first step is to search for an exact or at least for a generic match of a keyword or a combination of keywords.

However, we learnt very soon, that this does not fulfill all requirements.

There are at least two types of extended search methods desireable:

- retrieval by corresponding keywords and
- retrieval by synonyms.

Retrieval by corresponding keywords

Searching in unstructured data provides some challenges. Some of these can be addressed, when you can also find corresponding keywords. Some examples:

- In Europe we find a great variety in writing foreign names: For example, we were told that there are 8 ways to spell the name GHADHAFI.

- Misspelled words are also phenomena you should be aware of, like MITERRAND or MITTERAND instead of MITTERRAND.

- And in some languages we find many different word forms. The German word for house - HAUS, also occurs as HAUSE HAUSES HÄUSER and HÄUSERN.

There are algorithms that are able to recognize the corresponding words based on a factor for correspondence.

Retrieval by Synonyms

This means that retrieval covers a master-keyword and other keywords that have the same or a similar meaning, where in some cases a pattern match exists, in others not.

Some examples:

The above mentioned problem of spelling names is a case where some pattern correspondence exists.
But many times there is set of words, where this is not the case:

COMPUTER
DATA PROCESSING
DP
EDP

might be defined as synonyms.

In a multilingual environment you might define a keyword's translation as synonyms.
I.g. my country is called GERMANY here, but DEUTSCHLAND or ALEMANIA or TYSKLAND or ALLEMAGNE or ALEMANHA or GERMANIA in other parts of the world.

There are some problems associated with these techniques of extended searching:

- Synonyms must be defined before they can be used in document retrieval.
- Implementation is not just a trivial task.
- The biggest challenge, however, is performance: With the processors currently existing in the hp world (today is October 1983) response time might become a serious issue for three major reasons:

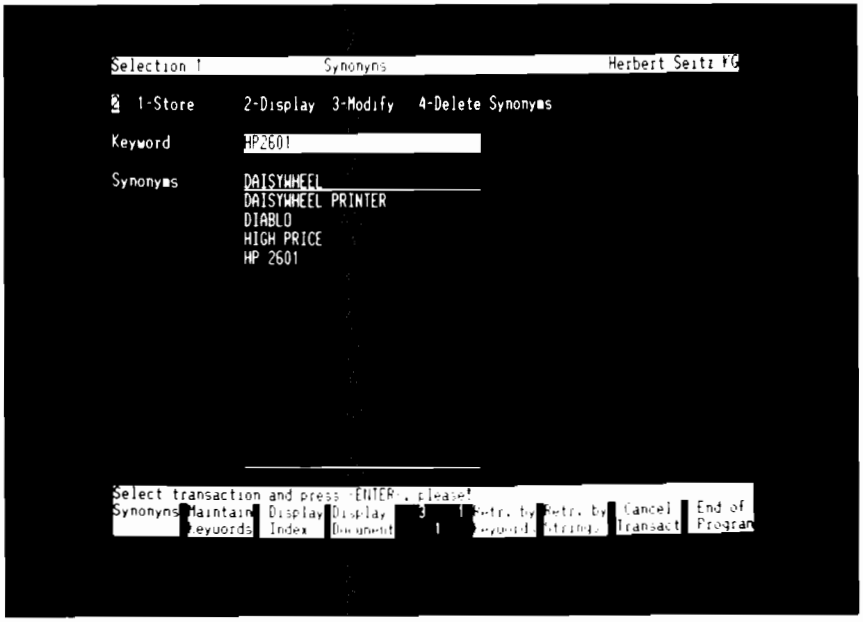
o some of the 3000 family's processors provide too little power for these non-trivial and non day-to-day data processing functions

o the number of disc accesses these systems can handle within a certain amount of time and the number of file-blocks that are present con-currently in main memory is limited

o mass storage systems with their mechanical access mechanisms are still relatively slow.

The tests we have done so far indicate that response times would be increasing in order of magnitude on today's computer systems if these techniques of enhanced searching would be fully implemented.

In order to overcome these shortcomings at least partially, we offer today online information about synonyms

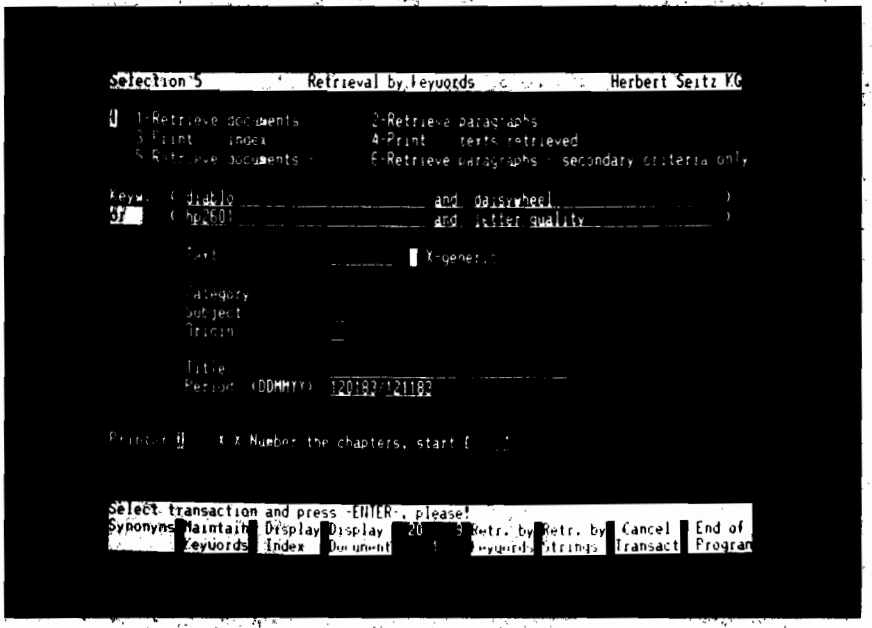


and similar keywords.

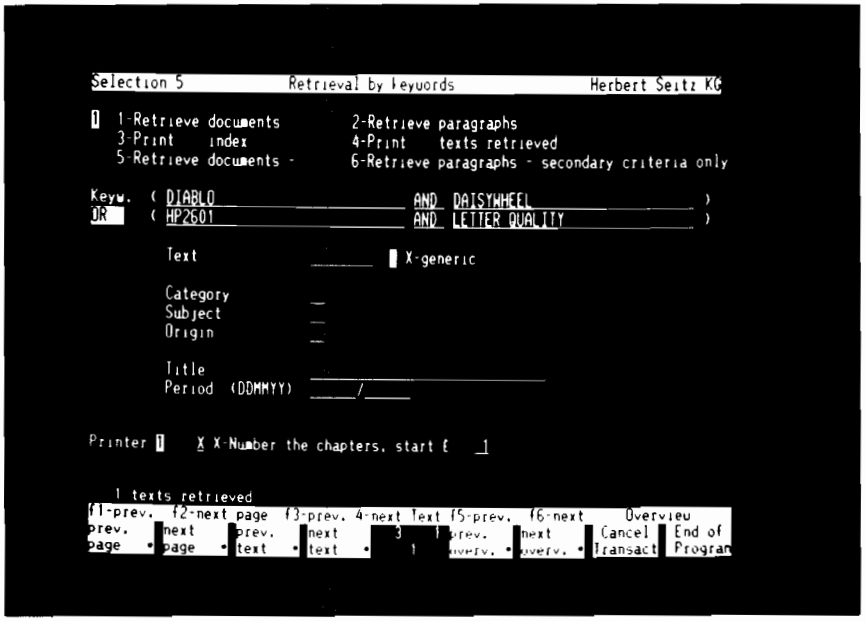
Reasonally efficient programming techniques for enhanced retrievals are already developed. We expect only a few more months (years?) waiting for faster mass storages and more powerful processors before these techniques can be made available in real life programs.

The User Interface - Retrieving documents

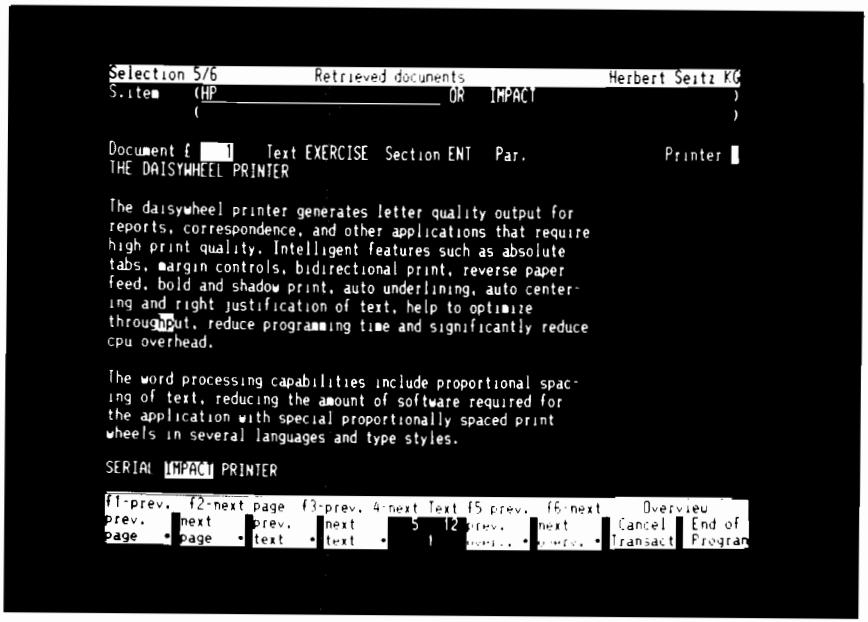
We selected a menudriven and softkey-oriented user interface. The user "fills in the blanks", all the information for a transaction at the same time, that is sent as a block to the computer. The user can define up to 5 primary keys plus secondary criteria. Primary keys are combined by the boolean AND / OR.



The number of retrieved documents and the number of retrieved strings are displayed together with the selection criteria.



Global function selection and fast browsing through the documents is controlled by softkeys, where no more than two levels of key labels exist.



Fast browsing thru documents means, that by pressing one function key, you should be able to display a list of retrieved documents. Just one more strike and you display the pages of a text. A handy print function should also be available. We designed an user interface which can be easily introduced to non-dp personnel. At our beta test site for instance, the system is used by journalists, that have very little practical experience with computers.

The User Interface - Thesaurus management

Of course, keyword index management should be done online only - and semi-automatically. That means it should be possible to define keywords with a minimum of work. We support two methods: first you can mark any word - or any pattern - within the text so the system proposes this pattern as a keyword. Second the system scans the text for all "words". In our system this is any combination of national alphabetic characters. Then it strips out all filler words of the current language in use, like in French LE LA ET OU and many more. After that, the words are displayed for manual modification and/or selection (each word only once - of course).

Again it is necessary - as for a word processor - to give full support to national character sets and languages.

By the way, it is an interesting side question how a "word" should be defined, if for example, a hyphen is part of a word or not. With current hp terminal hardware you run easily into trouble, because there exists no "Auxiliary Space" to define words as BUENOS AIRES and because only one type of hyphen exists where we would need two.

Of course, it must be possible not only to store, but also to modify, replace and delete keywords easily.

The User Interface - Customization

To make the system fit into a user environment, it is necessary to make it user customizable. Some of the features that should be configurable are

- Character sets
- Language used in documents
- Language in screens and messages
- Date format
- Printer assignment
- Text display format
- Sort criteria for retrieved documents.

The linguistic aspects are covered in more detail in "Edinburgh Proceedings, F.J. Boll - Linguistic aspects of word processing".

Internal Structures - Searching and Sorting

The hp 3000 allows efficient string handling via certain firmware instructions or indirectly via SPL. We would be very unhappy if SPL one day would become obsolete, because it allows almost optimal algorithms for scanning, searching, moving bytes, upshifting and so on. Especially we would like to mention that in SPL you can almost directly use instructions like

```
SCAN BYTES,
COMPARE BYTES,
MOVE BYTES with upshift.
```

Programming for unstructured data applications like word and text processing becomes more efficient - and in a certain way even easier - when using pointers and stacks directly, where one needs a minimum of instructions and a minimum of code.

Internal structures - Data Management

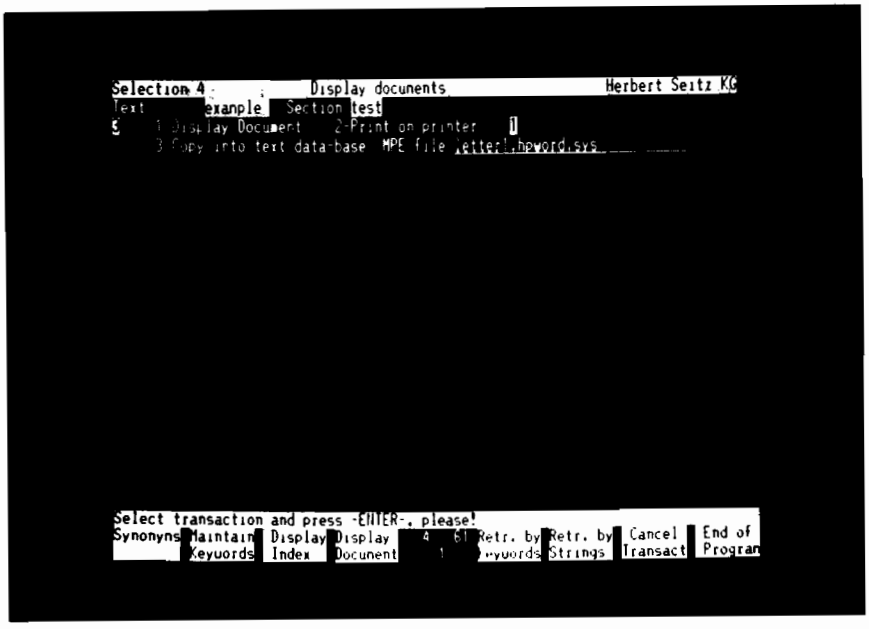
On user level you need a multi-level indexed data base for storing documents and also an indexed data base for the thesaurus.

As we wanted to focus our efforts on application problem solutions we decided to use an existing data management system. We realized, that KSAM has some of the structures we needed, inherently, IMAGE, however allows at least to emulate them against the user interface.

We realized that with both systems the interface might have had the same outlook, although the internal structures are so different.

And we choose IMAGE because we think that IMAGE is implemented in a much better way than KSAM, for example locking, the transaction monitor, access by a hp query language, and finally data integrity are items where IMAGE is by far better than KSAM. So we choose that file system although it meant some additional work to make it behave like an index sequential system.

In the beginning we designed our retrieval system as an enhancement to our integrated data- and word processing package **IDT**. We soon realized that we needed an Interface to MPE-Files in ASCII-Format. We wanted to be able to copy all kind of texts into our data base, like texts written by EDITOR, TDP, HP WORD or whatever. Thus this retrieval system can be used now by anybody having a hp 3000 computer, it is not restricted to a certain wordprocessor.The capability to re-convert printed documents (out of a library for example) into computer-processable ASCII format for our document data bases, is the next step in our plans.



References

Franz J. Boll, Linguistic aspects of word processing, 1983, Edinburgh Proceedings

Joachim Geffken, User friendly applications in commercial realtime data processing, 1981, Orlando Proceedings

Joachim Geffken (36) is resident of Bremen, West Germany. He is cofounder and part owner of the Herbert Seitz KG, an international software supplier and hp OEM.. Joachim Geffken has studied law as well as computer science at the University of Hamburg. He is Vice Chairman of the German National Users Group and has served on the IUG Board of Directors. Off business hours he works as a publisher and author. He also is actively involved in offshore sailing.

Franz-Josef Boll was born in 1951 in Mühlheim near Frankfurt - West Germany. After high school, he studied in Frankfurt to become a teacher for mathematics and social science. After one year as a teacher, F.J. Boll joined the computer community as a programmer and system's analyst. After two years he started his own dp-development company and designed the **IDT** word processing system, the **IDT-EF** retrieval program. He also does linguistic research.

His hobbies are playing piano and church organ. F. J. Boll is also a frequent traveller through South America.

STRESS CONTROL

by Dr. Simmons

About stress--there is some good news and some bad news. The bad news is that all of us have to contend with stressful circumstances and conditions every day of our lives. Some of us, it seems, have to contend with more than our fair share. There is even more bad news connected with this subject. There are personal and societal factors that make it difficult to cope with stress and leave us vulnerable to vicissitudes of life that take a heavy toll on our mental, physical, physiological and emotional well being. Before we look at these elements, let us say something about the good news.

The good news is that there is something we can do about stress in our lives. We can avoid some stressful situations and conditions and we can mitigate the effects of others by decisions we make and actions we take. This may be contrary to what some people think, but it is true, nonetheless. Let us now look at some of the conditions in our world that threat en our welfare. Generally speaking, our world gets more complex all of the time and complexities of life make it more and more difficult for us to live without a high degree of stress. Since its beginning, our country has undergone changes on a regular basis and it, as any living organism or entity, will continue to change as long as it exists. So, this will continue.

Our country began as an agrarian-rural society, went through an industrial-urban phase and now we find ourselves in a high techmegalopolis stage. The implications of this process are staggering. We find ourselves having to cope with stressful situations on all fronts. On one hand, it may appear to be a great thing to be in the middle of a high tech world. On the other hand, most of our population are not prepared to participate in it fully. If you are, and most who read this article will be, you are left to deal with those who are not. And that is no small chore. Much of our society finds itself in about the same situation as were the buggy makers during the first half of this century. The skills they have developed may be no longer marketable. Of necessity, they will have to make changes. For a mature person who has achieved a certain station in life and is expecting to continue an enjoyable life style, it can be a bewildering experience to find there is no job market for skills learned and perfected. Those who feel they can compete favorably in a high tech society will have to cope with the resentment of those who cannot and will have to assume responsibility for training and education of those who are in danger of being swallowed up by progress.

Not only do we have those who cannot participate in this new society, but those who find themselves in a precarious situation. Quantum leaps are being made in all scientific-technical areas and even those with a high degree of intelligence and a strong background in science-technology find themselves hardpressed to stay abreast of the ever in Competition mountain of knowledge that continues to grow each day. is keen and the burden it places on the individual to keep pace is dangerously heavy.

Technological advances are affecting everything we do in life. Communications, transportation, food growth, preparation and distribution, educational programs, all are undergoing sweeping changes. About the time we make peace with the idea of flying (many still have not), we have to cope with high speed surface

transportation. About the time we get comfortable with one computerized system, we have to learn to master another. And so it goes. Nothing remains the same. Our capacities for adjustment are taxed to the limit.

In addition to technological advances, we have to cope with the pressures of urbanized living in the extreme. We are finding ourselves with less and less space to call our own and more and more people with whom to share that space. Physical and emotional comfort are important and we need some space for that, just to get away from it all, as it were; but that isn't the end of it. The advent of the megalopolis on a large scale (by that, I mean more of them) has brought with it more and more demands on our abilities to relate to others and interact with a wide variety of personalities with diverse backgrounds who come from many different subcultures and who follow many different value systems. It is still difficult for a New Englander to deal with a Southwesterner and a Yankee to understand the ways of the Southerner. Yet, these people are being thrown together more and more in work situations which demand more and more of them in the way of understanding and cooperation. Not only are they being required to share the work place, but marriages are consummated between individuals who have not the slightest idea of how the other half lives, and has lived, and what their expectations of marriage are. It is certainly well accepted that those who marry have the greatest chance of success if they marry others with whom they share many facets of life. If they come from the same educational level, same socio-economic status, same religion, etc., their chances for marriage survival are better. With the new world, it is especially difficult to succeed in family planning and home building.

There are other societal factors that militate against successful stress control on the part of the individual. In a capitalistic society, Madison Avenue plays a very important role. Its role simply stated, is to make everyone of us unhappy with whatever it is we have to wear,to eat,to ride or to play with.

Buy something new is the clarion call that goes out to the entire populace. You simply cannot be caught dead in plain ol' Levis. You have to go with Calvin Klein, or Jordache, at least. You certainly don't want to invite friends to your home and have to get up and walk over to the TV and change stations. How primitive! So, we find ourselves really having to keep up with the Joneses and the Smiths and the Silversteins and the Delgados and the Wongs and Ferraris and the Slomchinskis and just about everybody else on the block and for blocks around. The grievous burden we bear as a result leaves us with unmanageable amounts of stress.

Not only does Madison Avenue dictate to us what we have to wear and what we should drive and the kind of recreation we should engage in, but it reminds continually of our shortcomings from day to day. Apparently, we all smell rather bad and have skins that are too dry or too oily (some of us have both, according to some recently revealed information) or hair that has split ends, or is too thick or too thin. And, of course, all of us are overweight so nothing can be done but join a weight watching organization that reminds us that we are totally unacceptable unless we follow their plan to the nth degree. It's a wonder that we have any sanity at all.

Madison Avenue doesn't stop there either. They bombard us with all kinds of sound advice. For example, one of the leading breweries has begun to advertise that you should "put a little weekend in your week". In other words, drink alcoholic beverages every day and do not let a day go by without getting in the proper amount of dissipation. After all, you wouldn't want your body to know how good it can feel if it is not basted with alcohol regularly. Another advertisement has to do with the TGIM club. Thank God, it's Monday! Now, when we wake up foundering from the excesses of the weekend, we do not have to suffer from sick feelings. Just start drinking again. Never mind trying to find out if somebody got the number of the truck that hit you.

Of course, Madison Avenue is joined by others who have a buck to make. As you drive down the street, any street, you will find advertisements on signs and directly on buildings proclaiming Happy Hour! And that's every day of the week. If ever there was a misnomer, it has to be that assertion. I have witnessed a considerable amount of embryonic misery watching people "enjoy" Happy Hour! It really doesn't matter though, because you can straighten everything out with another helping of Madison Avenue.

There is an ad running currently on TV and elsewhere that says essentially for you to go ahead and dissipate to your heart's content. Our tablet is designed for people like you--for those who burn the candle at both ends. The implication is that it is perfectly all right. You really don't have to worry about what you are doing to your person and to your interpersonal relationships and to your ability to work and be productive. Dissipate to your heart's content and then take one of our li'l ol' pills and everything will be lovely. And, of course, they make many sales as a result. It seems that the majority of people actually believe what they are told in this regard.

Feeling tired and worn out, lagging a little? Grab our brand of cigarettes. They refresh you. You will feel better, stronger, more sociable. You will satisfy that need to belong. Be one of the in group. It is rather easy to be victimized by this type of suggestion. Obviously, it works. Advertising pays; at least, it pays the advertiser.

We cannot limit ourselves in this discussion, however, to society as a whole. We must concern ourselves with individuals as well. There are some characteristics of persons that make it difficult for them to live lives that produce a minimal amount of stress. One thing that seems to stand out in the young is the God complex. I am immortal. Nothing will ever happen to me. I can abuse my body and my mind and I can defy the law of gravity and other physical laws and remain in complete control. Of course, there is a terrible price that society as a whole pays for this kind of thinking, but it seems impossible to counteract. It is bad enough to see this in the young, but when older people do not outgrow it, it is frightening.

Then, too, there are certain things that we feel we really ought to do; for example, "enjoy" the aforementioned Happy Hour. How droll it seems to consider going home when the workday is done. If you have any creativity at all about you, you will go where everybody else is going, to the bar. That's a sign of independence, of being our own person. How unique it would be to go home and play with the babies, or better yet, play with the babies' mama or papa. What a quaint idea!

There is another ritual that we must engage in. After a long, dreary winter when the sun has hid itself behind the clouds and over the hills for months, suddenly, we find ourselves surrounded by summer. The beaches are bedecked with natural beauties and we must hie ourselves away to the seashore for a little sun and surf. Of course, many of us come back with second degree burns, unable to work, perhaps lose our job, can't get close to anybody for two weeks just waiting for it all to heal so we can do it again. And many are the young ladies who worship the sun and damage their skins beyond repair because that is what everybody is doing. It doesn't make any real difference, of course, because Madison Avenue is advertising an emollient that will take care of everything. We will always have soft, alluring skin, no matter how much we abuse it.

It is amazing how many people choose to fight stress conditions with stress-producing agents such as alcohol and tobacco. Under the guise of relieving stress, these agents create and aggravate already stressful conditions. Not only do we use these damaging agents, but we deny our- selves much benefit from rest and relaxation when we use them during rest periods at work. Rest periods were introduced so people would be able to continue their work after a time of relaxation and recuperation. It is precisely at this time that most of us make a great effort to see that our bodies are denied this brief hiatus from stress by smoking or drinking stimulants which deplete the body's energy resources.

Human beings are also cursed with the belief that whatever happens that threatens well being can be handled by following the advice of a friend. interest in life, totally rested from the cares and worries of the every day world. In most cases, what will happen is that you will come back from such an outing with all of the worldly cares and stresses aggravated by tender feet, sore muscles and exhausted bodies. That is, unless you have been building up to such an adventure gradually. Stresses of life are not to be taken lightly, but require thought and planning to handle successfully. At the same time we are so eagerly following suggestions from our friends, and even casual acquaintances, for the improvement of our lot in life, we humans exhibit another peculiarity. We are quite willing and even anxious to pay professionals for advice not followed. Many unfilled prescriptions and cancelled checks will attest to this fact.

For example, we go to a doctor with a digestive complaint and we are not satisfied unless he gives us a prescription for some magical elixir that can relieve all of our strains and stresses. Of course, in many cases, we do not bother to have the prescription filled because by the time we get to the pharmacy, nature has done its work and we are already feeling a little better, or so we tell ourselves. Besides, when I told the doctor about eating highly seasoned pizza and drinking a little beer, he had the temerity to suggest that perhaps I should not eat highly seasoned pizza and drink beer. Perish the thought of modifying our behavior in order to decrease bodily stress. Everybody drinks beer and eats highly seasoned pizza. What would the pizza makers and breweries do for a living? Obviously, the doctor is not to be taken seriously. There is a little story about Babe Ruth that illustrates the frailty of human nature rather well. It is said of the Babe that he was a junk food addict before it was called junk food. On one occasion, he had an upset stomach during a ball game. Somebody asked him what he had eaten. His response was "four hot dogs, six cokes, three candy bars, some chips and an apple". After thinking a minute, he said he guessed he shouldn't have eaten that apple. I rather doubt that the apple was the culprit. Perhaps the story itself is apocryphal, but it certainly characterizes most of us in that we find it difficult to associate our behavior with the way we feel.

There is one other approach that we humans use to solve our stress problems that needs to be observed. Many of us really expend a lot of energy in trying to relieve ourselves of stress, energy that could be used wisely in any number of ways. This is exemplified by the experience related to me recently by a young lady whose husband is in the DP world. She actively pursues a career outside the home, also, and they decided their lives were so filled with stresses they thought were generated by urban living that they needed a place in the country to relax. Excellent idea!

So, they bought themselves a lot with a little cabin on it in a quiet country town north of Houston. Houston was where they labored day after day. Living in Houston is a real labor in itself. Anyway, they could hardly wait for the first weekend away from the city. Wild anticipation reigned. Great plans were made. Everything was going to be so perfect. Finally, Friday evening came and they jumped in the pre-packed auto and headed north on Interstate 45. Some three and a half hours later, they arrived at their little country retreat, battle weary from their struggle to negotiate the 60 miles. The return trip two days later required the same time and the same effort. Fortunately, they could see the fallacy of their thinking and sold their little dream place and learned to stay in their own back yard and really relax. So much for those factors that impede our search for relief from stress.

It is time to look at the good news again. We can exercise some control over our own lives and make for more relaxed and rewarding life styles that will result in less stress and displeasure with life in general. Now, this is sometimes doubted by individuals who feel they are victims of an uncaring world bent on doing them in totally. So, first we have to rid ourselves of a certain way of thinking and change our attitudes to a degree.

There is a tendency for many of us to blame our ills on something, or somebody, other than ourselves. We see ourselves as pawns in life's game of chess. We are being manipulated, used, abused and influenced in devious, diverse and myriad ways. We are victims of society, of our environment, anything except our own weaknesses and refusal to order our lives in such a way as to reap the most rewards, mentally, emotionally and physically. We obstinately cling to the idea that our adversity in life has nothing to do with our behavior. We simply do not have any control. That is the chief reason for the popularity of astrology in our time. I have had parents sit in my office and lament the fact that their child has seemingly abandoned standards and denied values held near and dear to the older generation. Instead of attempting to under understand what their offspring is actually doing and trying to say, it is easier to attribute it to the influence of the stars. We know our child is impulsive, irresponsible and inconsiderate but that is just the way Virgos are. (My apologies to all of you Virgos with sunny dispositions.) Since he is a Virgo, there is obviously nothing that can be done. Or, in the same vein, our child has a terrible temper. I guess it is true what they say about redheads. They are just hot-tempered. If they weren't, their hair wouldn't be red. It doesn't take children long to learn. They get the message quickly. It isn't my fault if I am socially unacceptable so I don't have to do anything about it. These children grow up to be obnoxious adults. There are grown people who will go into temper tantrums and embarrass and humiliate their friends, to say nothing of alienating them. Then, they make some remark to the effect that this is just the way I am. You will just have to get used to me. Not a thought is given to modifying behavior. This

disinclination to accept responsibility for ourselves seems to be bred into us at an early age and is resistant to extirpation.

So, what can be done? As I suggested, change your attitude, if that is necessary, and accept the fact that you can exert a great measure of control over your behavior and your life.

Once we get over the idea that we can find the answers to all of life's problems in pill or liquid form, and we understand that we produce most of our stress by our own thoughts and actions, we will be ready to go to work on improving our lot in life. Then, we will be prepared to do the following things:

1. Learn about yourself. This may involve seeking the help of a professional. There may be someone out there who can help you understand yourself better than you do at present. You will need to know about your intellectual ability and how able you will be to function on a level that requires profound and creative thought. For example, if you aspire to be the world's leading brain surgeon and you had trouble with intro- ductory Biology, pick another goal. It is a good thing to set goals, but they should be commensurate with your abilities.

You will need to know something about your personality and what traits play a dominant role in your interaction with others. If you have an opportunity to get a good position that requires constant contact with the public and you are basically shy and timid and averse to interaction with others, then, choose another position; or, if the position is appealing, work toward overcoming your weaknesses in this area. If you are a highly motivated individual who works at a fast pace, you probably would not enjoy finding yourself in a situation where you have to spend a lot of time cooperating with people who slow you down. If you are claustrophobic, do not take a position that confines you to an eight-foot square cubicle.

Your specific and general interests will have to be considered. If you have an enduring commitment to the Arts, do not place yourself in a position where you are surrounded by sports freaks.

As you find out about yourself, you will be forced to make decisions about what you see. You will have to decide whether you like what you see or whether you would like to change what you see. In any event, the information will be useful to you.

2. Learn about others. This applies especially to the persons with whom you have close associations. These are the people who can be supportive of you from day to day or who can cause you much grief if you cannot communicate well with them. You may not have any interest in participating in Christmas activities. You may not even want to send Christmas Greetings. It might not hurt, however, to consider the feelings of a mate. Your mate's spirits may be revived periodically by observing special days such as Christmas. It may give them an emotional lift and make them feel all is right with the world. As you learn about others, you can make concessions to them in order to have a more agreeable work place or home life. This does not mean that you have to abandon your own principles or make concessions contrary to your conscience. It simply means that you need to learn to consider the feelings of others when decisions are made that affect both of you.

You will find that the world will be a less stressful place if you learn to do this.

3. Learn about sources of stress. Sometimes, it is hard to accept the fact that stress can come from our own inability to handle disappointments such as the loss of a job or rejection by a mate. We tend to act as though all will be all right as soon as we have another job or another mate. We do not allow for the existence of emotional turmoil as a result of what happens to us. Sometimes, we cannot even recognize the source of our discomfort when it should be obvious.

In my files is the history of a 23-year old female student, apparently in good health, who ended up in a hospital for tests after a lengthy battle with fatigue. She went to bed tired, got up in the morning tired and was tired all day at work. It turned out that she had no business working in the atmosphere in which she found herself. She had nothing in common with her

co-workers. She was interested in the Arts and got most of her personal satisfaction from painting, visiting museums and attending dramatic presentations. Her co-workers were all scientifically inclined and had no interest in the things around which she wanted to build her life. It wasn't until she came to this realization that she was able to handle the problem in a rational way. Many of us do not want to admit that we have emotional needs and refuse to admit that any of our ailments could be psychogenetic. Do not allow yourself to think that you are above feeling the effects of stress due to psychological factors.

4. Realize what damage stress can do. Stress, not controlled, can have very damaging effects upon you physically, physiologically, emotionally and mentally. The list of ailments caused in part by psychologically stressful conditions is quite lengthy. Fatigue, constipation, tachycardia, chest pains, elevated blood pressure, insomnia, dizziness, diarrhea, headache, abdominal cramping and the list goes on and on.

In another of my patient files is the story of the young lady, age 30, who suffered from the stress of a divorce. She thought she was handling her situation with considerable aplomb, right up until the time the doctor diagnosed abdominal distress as ulcers. Of course, this was one of those cases in which the patient was "managing" her stress with cigarettes and alcohol. She remained anesthetized a great deal of the time but stress did its damage, anyway.

5. Learn to manage your time. Everything you do takes energy. It takes energy to be a good employee, even if you are also the employer. It takes energy to be a good mate. It takes energy to be a good parent, a good child. We only have so much energy to devote to all of the aspects of our lives. If we spend too much in one area, another area will suffer. If you don't pay attention to time allotments, you will end up exhausted trying to keep up with all of your commitments. We have to establish priorities and learn how to live with them.

6. Make a schedule. This is closely allied to time management. In the case of time management, we are looking at the time we have to spend. In the case of the schedule, we are looking at the number of things we have to do. It may well be that we will have to reduce the number of activities in which we engage. We may not be able to join a number of bowling leagues. It may be that we cannot handle the money for our favorite organization. We may have to turn down opportunities to learn more about our profession. We have to decide what endeavors really fulfill our needs as an individual and use our energies in pursuing them.

7. Plan recreation. This could be included under scheduling, but I think it is worthy of its own classification. Most of us engage in recreational activities when there isn't anything else to do. Our recreational activities are usually spur-of-the-moment decisions. We keep our calendar full of other things to do, for whatever reason. Sometimes it is purely psychological. We really don't feel that we deserve the time to play. There is so much to do and so little time to do it. At other times, it is simply a lack of planning. We just don't deem recreation worthy of attention. In order to control stress in our lives rather than having it control us, we need to take time for recreation on a regular basis.

8. Learn Relaxation Techniques. There are many techniques for relaxation. Deep-breathing exercises, deep muscle relaxation, stretching exercises, biofeedback, yoga, and meditation are a few. It is not within the scope of this paper to provide instruction in these methods. I am simply calling your attention to the fact that they are available to you. You already know this, but a little reminder may help. It will also be helpful if you will manage to incorporate your selected method on a regular basis and participate with a group in some instances. Any of the above-mentioned methods will decrease tension in your system, help you to sleep and rest better and make better use of your energies.

9. Learn to eat nutritious food. I say learn to eat nutritious food because that is what it takes, learning. It is easy to fall into habits and eating is for most people simply a matter of habit. Unfortunately, it is easy to develop a habit of eating those things that satisfy our palate rather than those that satisfy needs of the body. It is probably advisable not to try to eliminate all junk food from your diet at once if you are a junk food junkie. It would probably be an excellent idea to identify a few nutritious foods that you can live with and develop a taste for them. Then, go on to others while eliminating some of the least useful foods that you now eat.

10. Get a reasonable amount of exercise. I say a reasonable amount because people have a tendency to try to make up for 10 years of sedentary living in a two or three week period of intensive exercise. Usually, this does more harm than good. Walking is still recognized as one of the more healthful exercises and does not call for a great expenditure of money. Swimming is also excellent and it does not cost much either. Of course, if you are on an exercise program that takes up a great deal of your time and energy and it is suitable for you, that is good.

11. Give some thought to your interpersonal relationships. Ask yourself some questions. Are they satisfactory? Do I benefit from them? While you are about it, look at it from the standpoint of the other person. Sometimes people tell me they are not subject to stress problems. On occasion, I find myself suggesting that the people who are dependent on them for emotional support may have a great deal of stress due to their lack of concern. Usually, they agree with me. If you are not getting anything from a relationship, whether it's with your golf partner or your spouse, or whoever, is there anything that you can contribute to make it better? Be honest with yourself in making this assessment.

12. Work on reducing conflict in your life. I am speaking of conflict with others and also conflict of interests that you might have. For example, you may be interested in being well-informed and read so much that you do not have time to devote to other things that may be equally important to you from a business and social standpoint. You may need to spend more time engaging in social activities with other professionals. If you find yourself in constant conflict with others, whether at work or at home, this is draining you energywise and something needs to be done to protect you from damage to your health. If you need to make overtures to others to alleviate the situation, be sure to do so. To wait for the other party is just inviting more damage to yourself.

13. Financial planning. It is said that people get into more trouble over sex and financial matters (not necessarily in that order) than in any other area. Certainly, it is worthy of some consideration on your part. Most of us would like to have our cake and eat it, too, but few of us are able to do that. If we want to enjoy some of the finer things of life, we are going to have to forego some of the lesser things. Planning will have to be done with parties who have a stake in the results. Certainly, a mate should have input when financial decisions are being made.

14. Allow for plenty of rest to body and mind. As I have mentioned before, everything we do takes energy. The body and mind need time to replenish supplies of energy. There is no substitute for rest. Scientists still are not agreed on what takes place during sleep that is rejuvenating, but it matters little. The fact of the matter is it does rejuvenate, as does just resting from labors periodically, with or without sleep. Some of us take a lot of pride in burning the candle at both ends and working 16-hour days. Many of us take our pride right with us to an untimely demise, denying friends and family our companionship and association over a reasonable period of years. Also, you might find a little rest does wonders for your disposition. You might become more welcome to a friend or mate. The Good Book says that after God made the heavens and the earth, he rested. Enough said.

In concluding, let me reiterate that stress control is primarily a matter of self control and you owe it to yourself to control your life to the end that stress will not take control of it for you and take you through some miserable years before it deposits you in an early grave. The adoption of practices herein suggested will enable you to effectively combat stress and its ill effects.

DISC MEMORY CACHING ON A SERIES III WITH EXTRA DATA SEGMENTS

by KURT D. RAHM
SR. PROGRAMMER/ANAYLST
BOHEMIA, INC.

INTRODUCTION

A major bottleneck of any computer is disc I/O and the HP3000 is no exception. Hewlett-Packard has announced MPE V with disc memory caching which allows disc files to be resident in memory, therefore, providing increased throughput of disc I/O. Since disc memory caching will not be available on the Series III, users of these systems need some alternative to increase disc throughput. This paper explains how extra data segments, in conjunction with process handling and multiple rins, can be used to obtain the benefits of increased disc throughput without having disc memory caching.

DISC CACHING

Disc caching is a sub-system that utilizes excess main memory and processor capacity of certain HP3000's (Series 39, 42, 48 & 68) to eliminate a large portion of disc access delays. It locates, moves, and replaces disc domains in main memory for access by any user. A disc domain is all, or part, of a disc file.

When a process requests input from a disc file, a list of currently cached disc domains is searched. If the requested disc domain resides in main memory, a memory-to-memory transfer occurs between the cached disc domain and the requesting processes data area. The process will continue executing without an interrupt. If the data requested is not in main memory, a fetching strategy using numerous facts determines the optimal disc domain to bring into memory, then a memory-to- memory transfer occurs.

There are various options for disc caching when a write to disc is requested. The option selected is dependant on ; 1 - whether the required disc domain is currently cached, 2 - if there another write pending, and 3 - if disc caching enabled for writes. Assuming disc caching is enables for writes, a user should never have to wait for the physical write to disc.

With the addition of disc caching to a system, the delays to read or write disc data will be significantly reduced thus decreasing response time and increasing productivity. Disc caching, or an alternative technique, would therefore be a valuable asset to any system. You can get more information on disc caching from your SE, so I will not go into any more detail. This article will furhter discuss an alternative to disc caching with the use of extra data segments.

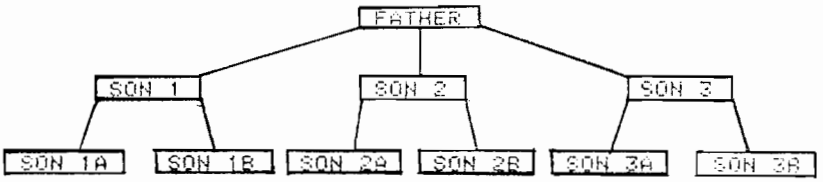
EXTRA DATA SEGMENTS.

An extra data segment, XDS, is a block of unstructured memory that is initialized to zero. The data in an XDS can be private, for the creator only, or shared by the creator and its sons (more later under process handling). Anything you use COBOL's workingstorage for, an XDS can also be used. Large arrays, table lookup, and parameter passing (linkage

section) are some examples. The question is then asked - If I can use working-storage or linkage section, why use extra data segments? Working-storage cannot be shared between processes, and what if your table is too large (i.e. MAXDATA can't be big enough). Extra data segments can be used to decrease an individual processes stack size, share data, decrease disc I/O, and increase throughput. Before we can look at how XDS's can be used to duplicate disc caching, we need to look at process handling (PH) and local rins.

PROCESS HANDLING

Process handling allows a user (process) to create, delete, activate, and suspend other processes. Process handling (PH) is required if you want to share an XDS. Refer to the family tree in Figure KDRTXT-1 for the following discussion. In order for a process to have access to an XDS, either it or its father or grandfather, etc, must have created the XDS. In order for 'SON 1A' to have access to an XDS, either 'SON 1A', 'SON 1' or 'FATHER' must have created the XDS. If 'SON 2' creates an XDS, 'SON 1A' doesn't have access to it, only 'SON 2', 'SON 2A' and 'SON 2B' would have access to this XDS. If you want more information about process handling, see chapter 7 in MPE Intrinsics manual.



LOCAL RINS

Local rins (resource identification numbers) are used to exclude simultaneous access of a resource by two or more processes within a family tree. Local rins can be locked, conditionally or unconditionally, and unlocked just like an IMAGE dataset, allowing exclusive resource access. If your application uses V/3000, you need some way to get the right screen to the right terminal at the right time. VOPENTERM defaults to $STDIN to paint the desired screen, but a parameter to VOPENTERM is 'termfile'. 'Termfile' can be file equated to some LDEV allowing the screen to be painted on that ldev and not $STDIN. By locking a local rin around the file equation and the VOPENTERM, some other process within the family tree can't change the file equation until the local rin is unlocked (assuming that all programs available to any process within the family tree locks the local rin prior to VOPENTERM). Local rins require MR capabilities and you now need to be careful to avoid deadlocking. Deadlocking occurs when two processes try to unconditionally lock a resource (local rin, data set, etc) that the other process has locked. Since MR allows multiple locks to be in affect, deadlock is a potential hazard. The only way to get out of a deadlock situation is to shutdown the system. If you want more information on local rins, see chapter 6 in the MPE Intrinsics manual. I want you to realize that only DS capabilities are required if you use extra data segments. PH is required if you intend to share extra data segments, and MR is required if local rins are required.

DUPLICATING DISC CACHING

Since data is either input from or output to disc I will handle these individually in presenting alternatives to disc caching. Since the majority of disc accesses are for input, let's discuss it first. Bohemia, Inc, has an Order/Entry system that uses 21 data sets to

edit orders and invoices for their sawmills, plywood and particleboard plants. These sets include information such as state, country, specie, grade, product, thickness, etc. Whenever an order/entry clerk enters an order, these sets are accessed to validate the entered data. Our procedure was to place these 21 sets into 3 XDS based upon record size and number of entries in each set. Each record consist of a dataset number, key, and miscellaneous data including a description. A batch job controls this whole process.

Referring to Figure KDRTXT-1, assume 'FATHER' is a program in the above batch job and therefore the key to the whole process. 'FATHER' creates the 3 XDS, sequentially reads the 21 data sets, loads the data into the approriate XDS and obtain a local rin so the right screen gets to the right terminal. Since the 'FATHER' has created the XDS and obtained the local rin, all processes it creates, or are created by his sons, have access to all 3 XDS and the local rin. The 'FATHER' then creates 5 sons (menus) on 5 different terminals, locking the local rin around the VOPENTERM. These menus later create a son based upon the option selected by the users, which will validate the input using the XDS's. If the value entered by the user is not in the XDS, the program goes to disc to attempt validation since the users have the capability to add items to the 21 data sets. In the Order/Entry system, the majority of disc input is a memory-to-memory transfer and not a physical disc access increasing throughput. For those who want more detail on how to create and access extra data segments, read about 'GETDSEG', 'DMOVIN' and 'DMOVOUT' intrinsics. For local rins, see 'GETLOCRIN', 'LOCKLOCRIN' and 'UNLOCKLOCRIN'.

This above mentioned process has been used at Bohemia since Jaunary, 1983, with great success. Our interactive users have commented on the noticable improvement in response time, and we have modified our printing of orders and invoices to access the XDS's for descriptions instead of going to disc.

POTENTIAL GOTCHA'S

Thinking through this process, realize there are a few potential problems in the above duplication of disc caching. The most critical involves the output phase. What happens if you get a system failure? The user thinks they have completed the transaction, but have they? If there is a system failure, there may be some data put by a user to the output XDS, but because of the volume or other delays, this data hasn't made it to the disc. If this is true, even if you have transaction logging enabled, no disc output has happened so your logging won't help and some data may be lost.

What about output to disc? One goal of disc caching was to decrease the delays of physical disc activity, thus allowing the user to proceed sooner. We have showed how these delays can be decreased for disc input, but what about disc output. This can be duplicated by adding another XDS to the 'FATHER' and another process. Instead of your programs doing 'DBPUT', 'DBUPDATE', or, 'DBDELETE', they output to the new XDS and let an output process do the physical disc activity. In order to do this, some information needs to be available to all processes involved in the family tree. The first record of this output XDS can contain the following: 1 - disc output displacement, 2 - user output to XDS displacement, 3 - length of XDS, and 4 - status flag. Displacement is the length (in words) from the beginning of the XDS to the beginning of the data to be transferred. Disc output displacement is the number of words to the beginning of the next XDS record to be put (updated or deleted) to disc. User output displacement is the number of words to the beginning of the next available XDS record for interactive users (i.e. where the next user is going to put his/hers next disc output detail). Since we will By using the XDS as a circular file, the length of the XDS must be obtained to determine the "end-of-file". When the end is reached process starts back at the beginning. The process flag can be used to indicate various status' of the output XDS such as XDS full. Records two through the end of the XDS contain the following: 1 - database name or number, 2 - dataset name or number, 3 - activity flag (0-free space, 1-add, 2-update, 3- delete), and 4 - data to be put, updated, or deleted. After the users have put the disc activity to the XDS, the output process can read the same data, and perform the desired activity. Since the interactive user is writing to memory and not disc, the delays associated with a physical write are not encountered and the user can continue processing sooner. A local rin can be used to control whether a user or the update process has exclusive access to the output XDS.

Security may also be a problem. Since the users at their terminals never get a colon prompt and never enter :HELLO, MPE's logon security won't work. Solution: create your own security that is used instead of MPE logon security.

One rule with process handling is - when the father stops, so does every other process in the tree. If something would happen to cause

'FATHER' to abort, all other process stop and the XDS's are lost. The only way this will happen with our 'FATHER' is with a "bounds violation" or "system failure".

The 'FATHER' is a batch job, and anything displayed to $STDLIST won't be seen until the job terminates. If a son aborts, we want to know the reason now, not hours later. SOLUTION: Redefine $STDLIST to a disc file and FCOPY the disc file to LP if you want to see it.

If you use local rins (MR capabilities), be aware of deadlock possibilities. SOLUTION: thoroughly test programs without local rins and MR capabilities (at more than 1 terminal at a time). Only after the program is tested should MR and local rins be added.

Kurt Rahm is the Senior Programmer/Analyst for Bohemia, Inc, in Eugene, Oregon. His work experience on the HP3000 family dates back to 1977, beginning on a Series II, to the Series III he now works on. Kurt has modified the Order/Entry system at Bohemia to access extra data segments, not disc files, to validate orders and invoices processed by their sales department. Kurt is an active member of the Oregon Regional Users Group (ORERUG) as a speaker and is a nominee for the presidents position of ORERUG.

HOW PRODUCTION CAN EXIST WITH DEVELOPMENT WITHOUT GETTING 'SMASHED'

by John D. Baker, P. Engineer
Hewlett-Packard

PURPOSE

This paper is intended for those in general management with relatively little experience with the HP3000. This is a description of an accounting structure and a naming convention that we developed at Kingston General Hospital. It is also a description of just one of several approaches that may be taken by management, some being more appropriate than others in certain situations.

Like everything else in the computer business, the development of an accounting structure is evolutionary and will change to meet the needs.

BACKGROUND

My background consists of being a Professional Engineer who got involved with running a very small computer shop for a hospital. I then moved on to health care consulting with Ernst & Whinney. I was very fortunate working at E & W in that I could work in health care as well as data processing. Part of the data processing requirement, however, was to assist the accountants with their audits where computers were part of the financial reporting.

Through this experience, I have encountered many methods whereby development takes place concurrently with production. In most instances, especially in the smaller shops, the process of implementing software relied on a "scouts honour" approach. Very little assurrance could be given to me that the user was aware that changes had or would be taken place. In many instances the user knew only after the users couldn't balance the books three weeks after a change was made to a program.

It is important from everyone's point of view (user, programmer, shareholder) that when programs are modified by data processing personnel, that all concerned are aware and that precautions are taken to minimize potential disasters that could take place.

Upon my arrival at Kingston General Hospital, I was greeted with the following:

* A series III with 1 MB memory and 240 MB disk.
* A financial system that crashed every month end and periodically during the month.
* No data processing plan.
* A potential law suit with the turnkey organization which supplied the hardware and software.
* Queen's University (who processed some of our systems, the major being payroll) was converting their hardware from Burrough's to IBM. I didn't know about this until late February. The conversion was set for May 1st.

I was also given one major mandate, besides putting everything in order. This mandate was to spread computerization throughout the Hospital. In short, turn around the bad impressions left by a bod implementation and make them positive.

This has been done, however, it has taken almost two years. The process we used was to:

* Establish a formal steering committee.
* Organize operations.
* Contain all new development.
* Establish standards such as:
  - o Database Standards
  - o Application Development Standards
  - o Documentation Standards
  - o Accounting Structure and Naming Conventions

I started on my treck and didn't get very far. My operations personnel consisted of basically two people who were not technically oriented. I had one programmer who had taught himself COBOL. He too didn't know much about HP systems. The Programmer's Course and the System Manager's Course weren't really all that helpful to me either. I knew what I wanted but I didn't have the knowledge nor the funds to purchase the knowledge. The next step was to read the System Manager's Manual over and over. I think I spent more time sleeping than I did reading.

I made a discovery last year. I wasn't alone. There were a few people at the IHPUG held in Montreal that had similar dilemmas to my own. It was there that I picked up more information than at any other single source except for the HP manuals. After Montreal portions of the manuals actually made sense.

INTRODUCTION

The primary purpose of a standard accounting structure and naming convention within a computer system is to provide:

* Access to as many users as possible.
* Data integrety.
* Data security.
* Meaningful file names for systems personnel.

A properly designed accounting structure protects a user's data from being modified by anyone except approved users, whether planned or accidental. The naming convention in conjunction with the accounting structure will allow the programmer to develop and/or modify program and data files using copies of live data without disturbing live data in any way.

ACCOUNTS

There are several account types assigned on the system. These are:

* SYS
  - o System Management
  - o Operating system and utilities
* HPSUP
  - o Hewlett Packard diagnostics
* PR0000
  - o HFMS production programs and run files
* QUEENS
  - o Programs and run files which were resident on the Queen's Computer System and which now reside on the HP
* KGH
  - o HFMS database, sequential files and internally developed files specifically related to HFMS
* DATABASE
  - o Database Manager's account providing maintenance and control of:
    - . Database schemas
    - . Data dictionaries
    - . System, application and program naming schemas
* LIBRARY
  - o Specifically designed to hold development tools to be used by programmers that can be accessed while in other accounts.

* PRODUCTION ACCOUNTS

  o Production programs and datafiles not specifically mentioned previously.

This paper is primarily devoted to detailing the specific accounting structure design of production accounts. The other accounts are designed for specific uses and in many instances, eg. HPSUP, the structure is dictated by other sources. In other cases, eg. DATABASE and LIBRARY, the actual structure will be evolutionary and will be entirely dependent upon the tools developed or purchased over the next few years.

FILENAMES

To understand the accounting structure, you must understand the approach MPE (HP's operating system) takes in naming files. A file name is defined as follows:

Filename.GROUP.ACCOUNT

This is a hierarchial structure. An account can have several groups and each group can have a multitude of files. Through this simple structure we can design a sophisticated accounting structure and naming convention that will allow us to meet our objectives.

The term accounting structure basically relates to the GROUP.ACCOUNT portion of a filename. The naming convention for programs and files relate to the Filename portion. The prime objective is to be able to have all files in one group within one account in order for the programmer to function without impacting live data.

PRODUCTION ACCOUNTS

All production accounts will be assigned the following groups:

GROUP	ACCESS	CONTENTS
PUB	Default	User's exclusive use
IMG	R,W,A,X,S,L:ANY	All databases for a specific account. Schemas and dictionaries to reside in the DATABASE account
SEQ	R,W,A,X,S,L:ANY	All sequential files
KSAM	R,W,A,X,S,L:ANY	All direct (KSAM) files
JOB	X:ANY;R,W,A,S,L :GU,AL	All job control (STREAM) files
DEV	R,W,A,L,S:GU,AL	All development programs, data, etc. Once tested and approved by the user, source, run files data, etc. can be transferred to the appropriate groups
USL	R,X:ANY;W,A,S,L :GU,AL	All production programs in object (non-exectuable) code
RUN	R,X:ANY;W,A,S,L :GU,AL	All production programs in executable code
SOURCE	R,W,A,X,S,L:GU,AL	All programs in source code for a specific account. Schema source and data dictionary source to reside in DATABASE
UDC	R,X,L:ANY;W,A,S :GU,AL	All account UDC's

```
DOC       R,X:ANY;W,A,S,L      All documentation related to
          :GU,AL               production runs
```

FILE SECURITY

There is an additional aspect to file ownership. This involves the use of user names. User names are assigned to accounts that are generic and relate basically to the function of a particular user. Passwords are assigned to the user. Passwords are NOT assigned to the account. By having passwords assigned to the account, all passwords would have to be changed on all JOB files and databases should anyone leave the Hospital. In the case of HFMS, this would mean approximately 40 man-hours to make the necessary changes. This is an impossible task given our present resources and software tools. By assigning a specific user name in a generic form, we can either change the password with relative ease or eliminate the user name from the accounting structure without fuss or bother. Our key concern is personnel having System Manager capabilities. This, however, will always be a concern no matter what security system is put in place.

USER NAMES

The following user names are assigned to each account and database designed by KGH:

USER NAME	CAPABILITY	HOME GROUP
MANAGER	Account Manager	PUB
MGR	Account Librarian	PUB
DATA0	Default	PUB
DATA1	Default	PUB
.	.	.
.	.	.
DATA9	Default	PUB
SUPER0	Default	PUB
.	.	.
.	.	.
PROG0	Account Librarian	DEV
.	.	.
.	.	.
PROG5	Account Librarian	DEV

Capabilities as defined in the above chart relate to capabilities within the account, not the programs that are being designed by Systems personnel. Specific capabilities within the actual application software are assigned within the databases and data dictionaries.

Each user is assigned to a Home Group meaning that every time a user sign's on to the system, he or she will have access to the files within that group. All users except PROG's are assigned to the PUB group. Applications designed by us are protected in other groups but are usable by all users of PUB.

All groups also have passwords assigned to them. All users have the capability to sign on to any group, however, should a user try to sign on to group other than their Home Group, the user must provide a password. Should the user sign on in the normal fashion, the user will be prompted for only the user password. When signing on to the system, a user does not have to provide a group password, unless the user is signing on to a group other than the Home Group assigned. In this fashion, we install hurdles for users who try to do things beyond their normal capability, yet virtually no difficulties are put in place when they meet the appropriate criteria.

Of course this approach (as do other password approaches) relies on the users to keep their passwords to themselves and not tell other people. This approach provides Information Services with the ability to change passwords randomly thereby increasing the security of each application. However, certain software tools are required to do this properly. With this approach 40 man- hours would not be required to change the passwords in each of the applications.

SYSTEM DEVELOPMENT

When a system is being developed by Information Services personnel, the programmer is assigned the user name PROG and an associated password. All development takes place in the DEV group of the user's account. Programs to be modified are copied to the DEV group as would data files, etc. Development takes place essentially withhout bother to normal users.

Once development or modifications have been completed, they should be tested by the users. This could, and should, take place within the DEV group. This is a situational call by the staff of Information Services. Once this stage is completed and the user has "signed-off" all programs, data files, run files, etc. these files will be transferred to the appropriate locations. The password for PROG will be changed, thereby providing the user with assurance that the account will not be modified without their knowledge and permission.

MAINTENANCE OF SOURCE CODE

All source code for each account resides in the group SOURCE. However, source code will not always reside on disk. This would tie up a key resource unnecessarily. Production source code is maintained on two tapes for each account. One tape is kept in the computer room, the other in the vault. The purpose of this is to ensure that parity errors that could occur with tapes would be minimized. Should one tape be unreadable, the second is available as a backup. There is a very low probability that both tapes would be unreadable.

Should a program need to be modified, the following sequence would take place. The program(s) would be:

* RESTORED to SOURCE.ACCOUNT
* Copied from SOURCE.ACCOUNT to DEV.ACCOUNT
* Renamed in SOURCE.ACCOUNT to version "1".
* Modified, tested and "signed-off"
* Copied from DEV.ACCOUNT to SOURCE.ACCOUNT as version "0".
* Purged from DEV.ACCOUNT

If this is a volitile account, ie. frequent changes are being made to source code, transferring to tape would only be done periodically. Whether the account is volitile or not doesn't matter, the process of off-loading source code to tape is the same.

Time is the only point of concern for volitile accounts. The process of off-loading source is as follows:

* STORE @.SOURCE.ACCOUNT to a scratch tape (precautionary step only)
* RESTORE @.SOURCE.ACCOUNT with KEEP option
* STORE @.SOURCE.ACCOUNT to computer room tape
* STORE @.SOURCE.ACCOUNT to vault tape
* Sign-on to the SOURCE group
* PURGE the group SOURCE

This last step will purge all files within the group without actually purging the group itself. If, however, the group is purged without being signed-on to the group, the group as well as all the files within the group will be purged.

NAMING CONVENTIONS

Generally the following convention applies to all file names whether they are program, data or run files. The exceptions such as databases, some Quasar products and library routines take the same basic approach.

FILENAME = SSAAnnTv.GROUP.ACCOUNT where:

SS = System:	The system or application alpha name, eg. AR for Accounts Receivable.
AA = Application:	The number assigned to the application or function or program series. This will start with "00".

nn = Number:	The sequenctial number applied to the application series, file, etc. This will start with "00".
T = Type:	The type of file this represents.
V = Version:	As the name implies, this is for version control. "0" is the current version - "1" indicates the most recent past edition - "2" the next most recent past editon, etc. We would normally not keep more than version 1 for archiving purposes. This policy, however, is determined by circumstances.

It is the responsibility of the Database Manager to assign, maintain and control file names to ensure integrity. The system and application names ie. "SS" and "AA" of the filename, are assigned by the Database Manager. To some extent, programmers will find it necessary to have license over the "AA" portion of the name. When this is the case, the initial "AA" is assigned by the Database Manager. Discretion must be used by the system designers when this control is required. The Database Manager is informed of all changes.

FILE TYPES

The type of files being identified are as follows:

T	FILE TYPE
S	COBOL source code
E	SPL source code
A	FORTRAN source code
C	BASIC source code
V	VIEW source code
F	VIEW fast forms file
Z	QUIZ source code
Y	QUIZ compiled code
K	QUICK source code
X	QUICK compiled code
G	QTP source code
H	QTP compiled code
D	Sequential data file
J	Job Streams
U	USL files (also see USE files)
P	Segmented executable code

Exceptions to the above rules are outlined below.

DATABASES

IMAGE databases are defined as follows:

SCHEMA:	SSAA.SOURCE.DATABASE
ROOT FILE:	SSAA00.IMG.ACCOUNT
DATA SETS:	SSAA00nn.IMG.ACCOUNT

QUASAR DICTIONARIES:

SOURCE:	SSQSAASv.SOURCE.DATABASE
COMPILED:	SSQSAADv.RUN.DATABASE

SS = System name

AA = Application name

S = Source code

D = Data code

nn = Sequential number assigned by IMAGE

v = Version

QS = Quasar product

QT: QTP

USE FILES

Use files used by system designers are maintained in the LIBRARY account in the group USE. The nominclature for use files is:

ISPPnnUv.USE.LIBRARY where:

IS: Information Services

PP: Program type

nn: Sequential number

U: File type representing Use

v: Version

Program types are defined as follows:

ED: EDITOR

QK: QUICK

QZ: QUIZ

COPYLIBS

The nominclature for copylibs follow the standards outlined below:

SYSTEM.COPY.LIBRARY

OR

SSCPYLIB.SOURCE.ACCOUNT

The file SYSTEM.COPY.LIBRARY contains all system related copylibs while SSCPYLIB contains all copylibs belonging to SS. Each module is named for convenience such as INVMST for Inventory Master. The great majority of the copylibs that are used to define record layouts are defined by the Database Manager by default in the design of the database and the Quasar schemas.

John D. Baker, P.Eng., is Director of Information Services at Kingston General Hospital located in Kingston, Ontario, Canada. His twenty years of professional experience include management consulting with big eight accounting firm, managing the systems engineering function at a large metropolitan hospital and work as a systems analyst and staff engineer. Mr. Baker has taught college level courses and is the author of numerous articles in professional journals. He holds degrees in both industrial and mechanical engineering and is a member of several engineering professional societies.

AUDITING THE HP3000 DATA CENTER

By Steve Finn, LCS Data Processing and
Betsy Leight, Project Resources

INTRODUCTION

The word "auditor" has unfortunate connotations. The data processing professional facing an internal audit, cannot help but have a vague unconscious vision of the IRS meticulously scrutinizing 1040 forms in a desperate search for illegal deductions.

In fact, the DP auditor is more correctly thought of as an efficiency expert. By analyzing the various aspects of the DP department from an objective viewpoint, the auditor can make reccommendations which will benifit both DP operations, and the company as a whole. This more benevolent image of the occupation is very popular with DP auditors, although it is admittedly unlikely to prove much of a comfort to those facing an audit.

The responsibilities of the DP auditor can range from questioning the integrity of an immensely complicated software system to telling some fool to get his coffee cup off the line printer. Here it must be understood that the auditor's concerns will vary with the auditor's expertise. While ideal DP auditors are as comfortable with the internal workings of an HP3000 as they are with basic accounting principles, ideal auditors are few and far between. But whatever the extent of technical knowledge, the most important qualification for effective DP auditing is nothing more complex than common sense, a systematic approach and some basis for comparison with other similar data processing environments.

As long as the auditor is not shy and not afraid to ask questions, all of the pertinent information will be available somewhere, regardless of the auditor's technical background.

Various and sundry articles on the auditing function have divided the concerns of the DP auditor into lists ranging from seven to sixty-five relevent topics. There are actually only two: Efficiency and Security. These two catagories do split into quite a few sub-sections, and the purpose of this paper is to provide a list of many of the concerns most often overlooked by the typical HP3000 shop.

In addition, we will focus specifically on the Data Center operations area rather than on such other areas as Systems and Programming, etc.

STANDARDS AND PROCEDURES

One area where the DP auditor can be particularly helpful is ensuring the existence and enforcement of proper Standards and Procedures. If everyone does things the same way, it's even money they're doing them right. Here is a checklist of reasonable auditing questions on Standards and Procedures.

Standards and procedures in the data processing center should include the following general topics:

Getting the right program in operation at the right time.

Inserting changes into programs at the right time.

Getting the correct data to the right program at the right time.

Protecting the data and programs from accidental or intentional destruction.

Determining that the data processed is complete and accurate.

Methods of physically moving inputs and outputs.

Procedures for controlling data, programs, and the flow of work.

Methods of scheduling work.

Methods of getting work rerun in the event of error or disaster.

Record keeping of work accomplished and resources available to do the work.

Determining that there are sufficient resources available to do the work.

Maintenance and other housekeeping associated with the operation of the computer center.

OPERATIONAL WORKFLOW and CONTROLS

Is input data from other departments complete and entered ontime?

Does the Data Center keep job accounting information?

Is it evaluated and used by management?

Is anyone notified in case of an production processing error?

Are errors documented?

Are error statistics accumulated or are they ignored?

Are errors followed up so that they will not reoccur?

Is downtime reported and are statistics maintained?

Is there a log of late reports and/or jobs?

Is there a formal communications channel between operations and other departments?

Is there a formal channel for communication of operations "tips" and "gotchas" to all operators?

Are problems that are encountered at the computer documented and is effective action taken to prevent recurrence?

Do operators get feedback on reported problems?

Are headers and SYSLIST information used and checked?

How is it known whether all reports and/or microfiche have been distributed to the to the proper user?

Have procedures been established to control distribution of sensitive output?

Do controls and procedures include the disposal of confidential reports when they are no longer required?

Are operator run instructions maintained up to date?

SCHEDULING

Efficient production scheduling is extremely important in providing a high level of reliability and predictability to the data processing operation. Ensuring efficient scheduling is an important funtion of a technical auditor. Here are some points to raise:

Are daily processing activities scheduled?

Is there a daily contingency schedule?

For batch production, are actual run times recorded? Is this data used to calculate expected run times for a given day? Are expected run times compared against actual to ensure that runs have not terminated abnormally?

Are nonscheduled run supported by a work request or other written authorization?

Are schedule deviations documented and followed up by a supervisor?

Is a firm PM schedule established and adhered to?

In an online environment, are user-submitted jobs recorded to allow forecasting of future schedules, resource requirements and special processing considerations?

Are ALL jobs submitted through or controlled by operations?

Does operations route all output to the appropriate destination or do users pickup their own output?

Are there standards for the type, quality, and quantity of forms kept on hand?

DATA SECURITY & ACCESS CONTROL

It may come as a surprise, but many successful businesses have information in their data base which should, for reasons unknown, be protected from loss and kept from the competition. Useful questions are as follows:

Are data processing employees instructed as to their responsibilities concerning confidential information?

Does management periodically review and update controls and security provisions relating to data?

Are live production programs physically separated from developmental programs?

Are all staff prohibited from running test programs against live files?

Are operations personnel denied access to sensative data files?

Are procedures in effect covering the acceptance and transcription of programs from development into production?

Are program library changes approved and accounted for?

Is acceptance testing of changed programs approved by operations before transcription to production libraries?

Does the approval include assurance that production documentation is also updated?

Are operators prohibited from renaming or transcribing programs without prior supervisory approval?

Are internal labels used for all data and programs files?

Are accounts, users and data files protected by passwords and lockword?

Are passwords and lockwords changed periodically?

Is an operations log maintained?

Is the area above the suspended ceiling in the computer room accessable only from that area?

Are blank checks and other negotiables:

Issued on run schedule basis only? Kept in a locked/secure area when unattended? Controlled by maintenance of access forms? Periodically inventoried?

Is the data center restricted from unauthorized access 24 hours a day, 7 days a week?

EQUIPMENT UTILIZATION and EFFICIENCY

One unit of measurement for DP efficiency is the U.S. dollar. Others include Swiss francs and German marks, but all of them are essentially monetary. Auditing is a business function, and business measures by money. So, when we say efficiency, we actually mean cost-efficiency. Sad, but true.

Once it has been determined that the entire data processing department is following a perfectly implemented set of standards and procedures to the letter, the auditor's attention can turn to efficient equipment utilization. This can be a particularly tricky area, especially if the auditor is an MBA with no previous computer experience. The key here is simply to ask questions openly about procedures and daily phenomena.

How much machine time is spent on reruns?

Are reruns analyzed? Are certain jobs expecially susceptible to reruns?

Are certain programs or jobs inefficient in the area of file design or utilization?

Is the full multiprogramming capability of the HP3000 being utilized for batch production?

Are mutltiple job streams run concurrently?

Are cpu-bound and I/O-bound jobs mixed to maximize overall throughput?

Are many jobs restartable without rerunning the entire job?

PERSONNEL UTILIZATION and EFFICIENCY

This is a sensitive subject. Wandering around the DP department with a clip-board and jotting notes while the operators discuss the football scores is unlikely to make the auditor many friends. And while management presumably wants to be kept informed, that guy who just sits staring at his terminal could be the Chairman of the Board's nephew. Here are some delicately phrased personnel questions.

Do operations personnel require extensive training and experience inorder to be effective in processing daily production work?

Do operators require extensive knowledge of each application they run?

Is there a system to schedule and monitor normal daily processing. Is the system effective? Does it operate without excessive manual involvement?

Do operators spend a large percentage of their time tracking jobs in execution, replying to program messages, changing the fences, etc?

Are operators required to modify JCL at run time?

Are all necessary tapes, forms and other resources available when needed or are they found in a panic?

Is the operations department treated as the stepchild of the data processing department? After all, the H3000 doesn't require and operator!

Is there excessive turnover? Why? Is daily production dependent on any specific individual(s)?

DISASTER PLANNING and RECOVERY

Is there an Emergency Plan that is adequate in relation to the risk?

Is it kept current?

Is is distributed on a "need-to-know" basis only?

Does the Plan include action for offsite storage of files and documentation?

Does the Plan specify:

The conditions for use of offsite processing? Application priority? Resource requirements? Job scheduling? Run documentation? Required tapes, forms, supplies, etc.?

Have formal written procedures been instituted for hardare backup?

ENVIRONMENT

Now here is something for the non-technical auditor to sink teeth into. It doesn't take a great deal of scientific expertise to mention that things might run a great deal more smoothly if people didn't have to climb over the disc drives to get to the water cooler. Here are a few suggested Environmental questions.

Is the workspace adequate? Are the facilities neat and orderly? Can all supplies be found quickly, when needed?

Are tape drives cleaned regularly? Are tapes certified regularly?

Are such auxiliary items as bursters and decollators located outside the computer room, but in line with the flow of work for the department?

Are tapes, discs, etc., stored in a closed, fire protected, and limited access area?

ASSISTING THE AUDITOR

The best way to assist a DP auditor, assuming you are so inclined, is to provide as much information as possible. This can be done by studying the lists of suggested questions and answering them in advance. But a more efficient method of providing the auditor with information is to implement a software system which leaves clearly defined audit trails and which issues a wide variety of reports, so that valuable time is not wasted ferreting out information. In fact, perhaps the first suggestion of each DP audit report should be the installation of a better system to gather the proper data and to make the job simpler next time.

(More extensive checklists will be distributed at the presentation.)

Steve Finn, Corporate Treasurer of LCS, Inc.

Mr. Finn is an experienced Data Processing Auditor, with 11 years of background in the Financial and Computer fields. A graduate of San Jose State University, he took his MBA from the University of Santa Clara, and later taught accounting at Canada College.

Mr. Finn spent 4 years as a Mrketing Director at TYMSHARE, INC., and currently serves as Corporate Treasurer of LCS, Inc., a large Stock Transfer Agency.

He is an active member of the Western Stock Transfer Association and a former President of the Data Processing Management Association of Greater San Jose.

Betsy Leight, Vice-President of Project Resources

Prior to founding PRI, Mrs. Leight spent four years as a Systems Analyst with Hewlett-Packard Company, where she was responsible for the installation and support of HP3000 systems throughout the North Western United States.

Mrs. Leight worked at IBM Canada and New York for seven years as a Pro- grammer, Systems Engineer, Instructor and Manager where she managed over 30 programmers developing major business application systems.

Mrs. Leight studied Computer Science at Boston University and Carnegie Institute of Technology. She has attended numerous technical and man- agement training courses at IBM and Hewlett-Packard as well as at other independent training organizations.

MICROCOMPUTERS AND THE HP 3000: AN ISSUE OF COMMUNICATION

R. GREGORY STEPHENS
Lawrence Livermore National Laboratory,

INTRODUCTION

The central theme of this paper is the consideration of communication issues important to the HP 3000 user selecting a microcomputer to be used as an integral part of their HP 3000 environment. The main issues that will be explored are 1) terminal emulation, including V/PLUS compatability and HP 2622 emulation, and 2) ASCII and BINARY file transfer. Specific references to the HP 86/87, HP 120/125, HP 150, HP 200, Direct 1025, Direct 1625, and IBM P.C. computers and the extent to which these machines meet the needs of HP 3000 users will be considered.

The issues involved in selecting a microcomputer are as numerous as user applications, but one issue which is crucial to HP 3000 users is communications. I will address two microcomputer communications issues which the available hardware/software products have address to varying extents. These issues are terminal emulation ad file transfer capabilities.

Most microcomputers can communicate with the HP 3000 as a dumb teletype style terminal with available communications software or user written software and many can download ASCII files. I have therefore chosen to review microcomputers which have expanded capabilities beyond these (i.e. VPLUS/3000 block mode communication and/or binary file transfer).

Because of time constraints and hardware/software availability I have not reviewed the HPCOM by System Automation Corporation of Silver Spring, Maryland. HPCOM is an HP 26XX emulation program for the Convergent Technologies microcomputer. Alleyn-Day International of Berkeley, California produces Personal-VIEW; a VPLUS/3000 compatible terminal emulator for CP/M based microcomputers.

HP 86/87

TERMINAL EMULATION

The HP series 80 microcomputers are the oldest of the microcomputer systems which were evaluated. This factor contributed to some communications limitations which are discussed below. The 80 Series uses a HP proprietary processor and operating system in ROM. HP 2622 emulation is provided by HP through its' TERM80 program. This program requires a Z80 CP/M card and an RS232 interface card which may be purchased optionally.

With this hardware/firmware/software package the user can almost emulate an HP 2622. One problem is that the HP 86/87 screen displays only 24 lines. Because the HP 2622 has 26 lines, 24 display lines and 2 lines for function key labels, TERM80 allows you to toggle the function key displays on and off. When executing VPLUS/3000 applications which use the function keys this toggling becomes tiring. In character mode with the function keys displayed and the prompt on the 24th line you can't see what you are typing because the function key labels override the normal screen.

I ran several non-VPLUS block mode programs such as QEDIT, and HPToolset without any errors. However, the only display enhancement supported is inverse video. Half-bright, underline, blinking, and security enhancements do not work. The line drawing set available for the HP 262X terminals is not supported with the HP 80 terminal emulation. While the baud rate is configurable at 110 to 9600 bps the actual maximum speed is less.

Although the configuration screens for TERM80 are very similar to the HP 2622 terminal the keyboard is significantly different. There are several important keys which are not on the HP Series 80 keyboard which require the user to remember the double keystroke used to implement the break key, escape key, enter key, aids, modes, and user keys. Several of the editing keys also require double keystrokes and on a few occasions I found myself deleting lines instead of characters while I was becoming familiar with the keyboard.

All keys are typeamatic and when holding down a key it was displayed nearly as fast as the 262X terminals do except when the function keys were displayed. In that case the character was repeated much slower as the function key labels were being flickered on and off.

FILE TRANSFER

File transfer is supported through the Data Communications Pac. This program runs under the HP 80s' native operating mode. One of the first limitations this poses is that files transferred to the HP 80 cannot be used with CP/M applications since the disk formats for the two systems are different.

The transferred files are created with a file type of DATA and with 256 byte records. When I listed these files with the Data Communications Pac, carriage returns and line feeds were not interpreted which caused the records to be wrapped around the screen.The records were not easily readable and I was unable to edit them since files are created in native mode.

The Data Communications Pac may also be used as an asynchronous terminal emulator. This allows configuration of an external printer which may be logged to, baud rates configurable to 96<0 bps (actually runs much slower), DC1/DC3, ENQ/ACK protocol, and error checking.

HP 120/125

TERMINAL EMULATION

The HP 120/125 comes with Block Format/125. This program allows the user to run most VPLUS/3000 applications. The problem with this program and the HP 120/125s' terminal emulation in general is that the terminal emulation code is not the same as a standard HP 2622 terminal and it is not stored in ROM.

Because it does not emulate a standard terminal, some third party software packages and HP software packages will not run. I tested this program using QEDIT by Robelle, HPToolset, and HPSlate. HPSlate required me to select a supported terminal since the HP 125 is notrecognized, it allowed me to continue, and finally told me I had only 4K of terminal memory and that 12K was required to run HPSlate. While the HP 125 has approximately 9400 bytes of screen memory (more than twice that of the HP 2622) when Block Format/125 is loaded the available memory is reduced to about 3760 bytes of memory, less than the HP 2622. The only full-screen editor I was able to use with the HP 120/125 was TDP. (Using TDP I specified that I was using an HP 2622.)

Block Format/125 only allows one display enhancement per line. When Block Format/125 is loaded it traps the communication and interprets the escape codes required. Because of this implementation, communication with the HP 3000 is significantly slower when Block Format is loaded than it is when the communications code in ROM is used. The code in ROM also does a better job as a terminal in that all

four display enhancements are interpreted correctly and any combination of the four enhancements may be displayed on any given line of the screen.

Another problem inherent in RAM-based block mode emulation is that if at any time you wish to run an application on the HP 120/125, such as Series 100/DSN/Link, you must re-load Block Format/125. This becomes a time consuming when you are alternating between applications having to re-load Block Format each time.HP has alleviated this problem to a certain extent by allowing you to load the block format code from the HP 3000. Using this method the user runs a program on the HP 3000 which downloads the block format code, located in data files BFDATAA.PUB.SYS and BFDATAB.PUB.SYS, into the HP 120/125.

The HP 120/125 has the same screen display and keyboard as the HP 2622.It differs in its' configuration menu and options.The HP 120/125 does not have the memory lock feature. The MODES keys are the same as those on the HP 2622 terminal with two exceptions. The function key normally used for the memory lock key stores a 'LOAD OP SYS' key which is used to switch between remote mode and local CP/M mode to load the CP/M operating system.There is also a 'LOCAL OP SYS' key in place of the standard block mode key. This key toggles in and out of the CP/M operating system mode. The line drawing set available on the HP 262x terminals is not available on the HP 120/125, nor can you use any display graphics. The arrow keys cause you to scroll through terminal memory when you attempt to move beyond the top or bottom of the screen instead of wrapping around the screen as the HP 262X terminals do.

FILE TRANSFER

File transfer on the HP 120/125 is provided by the Series 100/DSN/Link program. This program supports the transfer of ASCII and BINARY files between the HP 3000 and the HP 120/125. It also allows the user to log his session to a local printer and/or disc file.Files with record sizes of up to 512 bytes may be transferred. Files up to the storage capacity of the disk drive may be downloaded to the HP 120/125.

To support binary communication with the HP 3000 DSN/Link is supplied with a program for the HP 3000, LINK100.PUB.SYS, which the user must transfer to the HP 3000 upon installation of DSN/Link. When uploading HP 3000 program files which are stored on the HP 120/125 you must set the record size in DSN/Link to 256 since this field is specified in bytes. Once the file has been uploaded the user must re-copy the transferred PROG file into the proper HP 3000 PROG file format. The Series 100/DSN/Link manual does not document these commands. To execute the program DEMOPROG having uploaded it to the HP 3000 from the HP 120/125 you must issue the following MPE commands:

```
:RENAME DEMOPROG,TEMPPROG
:FILE DEMOPROG;CODE=PROG;DISC=,1;REC=128,1,F,BINARY
:FCOPY FROM=TEMPPROG;TO=DEMOPROG;NEW
:PURGE TEMPPROG
```

One problem I have encountered in transferring files is the termtype specified on the port you are using to access the HP 3000. Some modem ports on the HP 3000 are set to termtype 9, allowing non-ENQ/ACK terminal access. When attempting to transfer files on ports with termtype 9 I have received a "WAITING FOR HOST RESPONSE" message which would not go away. To solve this problem I set the termtype to 10 (Note: this is not documented in the Series 100/DSN/Link manual and may have been corrected in a newer version of the software). Although BINARY file transfer with error checking is supported, I have had reliability problems when transferring binary files over standard telephone lines (most likely due to bad phone lines).

Series 100/DSN/Link also allows BINARY and ASCII file transfer between other Series 100 microcomputers including the HP 150.

TERMINAL EMULATION

Of all of the microcomputers I have reviewed the HP 150 is the most sophisticated in terms of communication with the HP 3000. It emulates the HP 2623 graphics terminal including full VPLUS/3000 compatability and tektronix graphics The HP 150 also has enhanced capabilities beyond those of the HP 2623.

One of these enhanced capabilities is the security field enhancement which allows specification of a field in which the characters typed are transmitted but not displayed on the screen. The HP 150 also has a smooth scrolling feature.

The other obvious enhancement is HPTouch. This is a powerful feature which may be accessed from the HP 3000 through a set of escape code sequences.One problem with this is documentation of these sequences. The HP 150 is supplied with a terminal reference manual and a users guide. The HP 150 terminal reference manual documents the terminal capabilities including a clear but incomplete section on the escape code sequences. It would be appropriate for the touch screen escape code sequencies to be documented in this section but they are unreferenced.

The documentation on the escape code sequences which I did receive from the HP 150 PICS was incomplete and in conflict with the escape code sequences used in the BASIC program in the October 1983 issue of BYTE magazine. Documentation concerning the I/O system has also not been available so far.

The HP 150 has a very nice keyboard which may become a HP standard. The keys have a good touch and the roll-over is fast. One problem I encountered was the placement of the escape key next to the left-hand shift key. The more casual user may encounter 'enhanced' problems by accidently hitting escape key while typing. The new keyboard does not have AIDS or MODES keys and uses instead several combinations of the shift, control, user, and menu keys to arrive at the same results. You can scroll through approximately two pages of display memory. You cannot, unfortunately, use the standard 256K of RAM as display memory.

FILE TRANSFER

File transfer is facilitated through Series 100/DSN/Link for the HP 150. This program functions the same as the the HP 120/125 version of Series 100/DSN/Link and is supplied with a new version of LINK100 for the HP 3000. LINK100 must be installed on the HP 3000, replacing any existing version of this program.

HP 200

TERMINAL EMULATION

The HP 200 has one terminal emulation package at the time of this writing. This package is the Asynchronous Terminal Emulator and itis a subset of theHP 2621/non-block mode terminal. There will soon be available an HP 2623/block-mode graphics terminal emulator which will address the concerns of the HP 3000 users.

The new HP 2623 terminal emulator will most likely require at least 512K bytes of RAM. It will not support the line drawing set, display enhancements, hard reset, self-test, nor frame rate. This emulator will also only display the top line of the function key labels since there are only 25 lines on the 9816 display.

FILE TRANSFER

The Asynchronous Terminal Emulator (ATE) also allows ASCII file transfer between the HP 200 and the HP 3000.One nice feature is when the user mis-types a command line it allows him to position the cursor back on the line in error, correct the line, deleting the colon prompr, and press the execute key to retransmit that line. The ATE is supplied with an environment disc which contains portions of the Pascal operating system, I/O configuration, and other Pascal system and library routines required to run the application.

The new HP 2623 emulator will also support ASCII file transfer. Neither the ATE nor the new emulator will support BINARY file transfer.

DIRECT 1025/1625

TERMINAL EMULATION

Although the Direct 1025 and Direct 1625 have differing CPUs' and operating systems (Z80/CP/M and 8088/MS-DOS) Direct has informed me that they are functionally equivalent in their communications capabilities.Due to hardware/software availability I was unable to review the Direct 1625 but the comments concerning the Direct 1025 should apply to the Direct 1625 as well.

Both systems support emulation of the HP 2622 and HP 2645A terminals. This is implemented in ROM and is easily selectable through the setup/configuration screen.

The Direct 1025 has one setup screen which incorporates both the terminal configuration options and data communications options which are standard on the HP 262X terminals. The user may select baud rates between 50 and 19.2K bps.620 bytes of RAM are available as a buffer for the communications port or as a typeahead buffer. The user can select between 240 and 620 bytes of this memory to be used for this buffer the remainder of which will be used as a type-ahead buffer in local CP/M mode.

Direct also allows specification of a scroll rate through the setup screen. The three options available are JUMP, SMOOTH, and SLOW. The jump option displays characters to the screen as they are received. The SMOOTH and SLOW rates both provide easy to read smooth scrolling at two speeds.

Directs' implementation of VPLUS/3000 block mode is very good with some minor Exceptions. When developing systems for non- technical users you do not usually want to allow them access to the configuration screen or modes keys. VPLUS/3000 disables the AIDS and MODES keys by default but the Direct 1025 still allows access to their setup screen so that the user could alter his configuration. Access to the terminal test, block mode, and format mode keys is also allowed. All of the display enhancements worked but I encountered one problem with the security enhancement. Security fields defined in a VPLUS form did not work but the escape code sequence for the security enhacement did work when I was in character mode.

I have had no problems with applications using several combinations of the block mode, format mode, and line/page mode. These included HPToolset, QEDIT, and TDP.The only application I found which had problems was HPSlate. This problem was a firmware bug which Direct responded to promptly by calling my local service representative.

One of the most convenient features which Direct provides that none of the other manufacturers provide is viewing 132 characters per line on the screen at one time. This feature can be extremely valuable to programmers coding and testing reports and to users who wish to view existing reports on-line instead of waiting for batch reporting schedules. Use of the 132 character line is nicely implemented so that the user can toggle between 80 and 132 characters with a keystroke.

One drawback to the Direct keyboard is that a special function key must be used like a control key to perform the standard HP editing functions including the break key. The keyboard has 86 keys, including a numeric keypad, which also performs editing functions, and including the eight standard HP 262X function keys.In comparison, the HP 262X keyboard has 100 keys including numeric keypad.

FILE TRANSFER

Direct allows transfer of ASCII and BINARY files between the Direct 1025 and the HP 3000 via their Link/1025 program. The Link/1025 package comes with two programs. One program runs on the Direct 1025 and the other runs on the HP 3000. When you run Link/1025, it checks to see if the group you are logged onto has the Link/1025 sister program, an SPL program that runs on the HP 3000.If the sister program is not in that group, it uploads the program from the Directand begins transferring the file.

LINK/1025 worked well with no problems and I was pleased that it automatically determined whether its' sister program was needed and transferred if required. Because the program resides in the users group I was able to begin using the package earlier that other packages which require the system manager to install the program.

IBM PC

TERMINAL EMULATION

Having had numerous requests for access to our HP 3000 system from users with IBM PCs', I have reviewed as many block mode terminal emulating programs for the IBM PC as possible. The programs which I have reviewed include; PC2622 by Walker Richer & Quinn, Inc., Seattle, Washington; PCCOM marketed by DWJ Associates, Inc. of Ridgewood, New Jersey; and VDTE: Video Display Terminal Emulator by Inner Loop Software of Los Angeles.

The IBM hardware has some physical limitations which each vendor had to account for. The IBM screen displays 25 lines while the HP 262X terminals use 24 lines plus 2 lines for the function key labels for a total of 26 lines.Each of the vendors accounts for this descrepancy in different manners. The other limitation is the keyboard. Since there are significant differences in the placement of keys, a logical easy to to understand mapping of these keys is important for ease of use.

PC2622 initially displays a help screen which lists the keys used to display the AIDS, MODES, and USER keys, including the HP 262X special keys and there equivalent on the IBM keyboard. One of the important differences between this program and the others is initially apparent. The difference is that there are AIDS, MODES, and USER keys which are similar to those on the HP 262X terminals and that when VPLUS/3000 disables these keys (by default) access is correctly disabled on the IBM PC. The other vendors do not support this.

PC2622 allows the user to save differing configurations in separate configuration files which will also store pre-defined user keys. This allows the user to store HELLO commands and if you have a built in DC Hayes Smartmodem you can store a phone numbers in the function keys and automatically dial a computer with the stroke of a function key.

The configuration is flexble and easy to use. The baud rate is selectable between 110 bps and 19.2K bps but the performance is as expected less than the selected rate. The device control specifications are more flexible that the HP 2622 since they allow the user to send data to an external printer, remote system, or to disk.

One nice feature which PC2622 has that the others don't is a user configurable record width. This allows the user to scan files with record sizes over 400 bytes viewing 80 characters at a time.The user scrolls horizontally in either direction with one keystroke.

To resolve the shortage of one disslay line on the PC the user may toggle the function key labels between two lines, which displays over the 24th line (you can still scroll the screen up a line to view the 24th line), and one line, which displays all 24 lines and the first of the two function key label lines, and no function key labels. This solution is flexible and the labels are toggled quickly.

All of the display enhancements worked with the exception of underline which is substituted by a yellow background on color screen monitors. Included in the enhancements supported was the security enhancement which is not supported on the HP 2622 but did work with PC 2622. The display functions enhancement is supported but it displays a graphics character set instead of the abbreviations displayed by HP. This program is also one of the few which supports the line drawing set. As far as I could tell PC2622 used all of the availble RAM as display memory which allows for more memory than any of the terminal emulators.

I successfully executed several full screen editors including QEDIT 3.1, QEDIT 3.1.8, HPSlate, and TDP.HPToolset locked up when it attemted to load the softkeys. (I found this package exceptionally well designed and it surprised me in several of its capabilities.) Complete, well-written documentation would be welcome.

PCCOM by David McGrew is stated to be a "HP 2382 terminal emulator".However, it did have some problems.None of the full screen editors I tested, including HPToolset, HPSlate, TDP, QEDIT 3.1, and QEDIT 3.1.8, worked dor did FORMSPEC or ENTRY. The formspec fields, and all fields defined in inverse video, allowed only one character on input in each field.On all of the block mode applications I ran the function keys were being ignored almost completly (attempting to get any of the softkeys to work I did lock the keyboard several times).Security fields, not supported on all block mode terminals, did not work using PCCOM, although they did work on the other two products for the IBM PC.

There appear to be serious problems in PCCOM's emulation of the HP 2382, the most frustrating of whichwas locking up thU IBM P.C. and lose of my session on the HP 3000 several times while attempting to use HP 2382 compatible software products.Repeated soft resets would not unlock the keyboard nor could I break out of the program since the break key is not implemented.I finally had to turn the

IBM off and execute PC2622 to break out of my locked session with the HP 3000.On several occasions my line to the HP 3000 had been disconnected.

The third product I tested was VDTE (Video Display Terminal Emulator) which is marketed as a VT52, HP 2648, and HP 2624 terminal emulator by Inner Loop Software. All of the testing I conducted in HP 2624 mode.This program has the least similarity to the HP 262x terminals.The configuration options are accessed through a main menu which is available at any time, whether the AIDS, MODES, and USER keys have been locked or not.

VDTE would not execute any of the above mentioned full screen editors but would run my VPLUS/3000 applications including FORMSPEC. It also supported the security field enhacement.Not all combinations of the display enhacements worked but in every case if a field was specified with security on this enhancment worked.

FILE TRANSFER

Each of the programs supported ASCII file transfer by means of logging. The only product which supported a true file transfer facility, in terms of ASCII and BINARY file transfer with error checking and retry, was PC 2622.

Like Series 100 DSN/LINK PC 2622 comes with a sister program for file transfer which resides on the HP 3000. Walker Richter and Quinns' implementation is slightly more flexible in that the user may select where this program resides on the HP 3000 instead of requiring installation by the system manager. PC 2622 displayed a status screen with information similar to that of DSN/LINK. A display of the number of characters/records in the file and the current number transferred is updated regularly. An optional error status may be selected which maintains information concerning several types of errors which may occur in transmission including timeouts andparity errors.

CONCLUSION

Having reviewed all of the microcomputers available which have communuication capabilites beyond asynchronous terminal emulation, the systems which stand out as exceptional in their interface with the HP 3000 are those which were clearly designed with this specific capability in mind.

The HP 150 and DIRECT 1025/1625 microcomputers clearly stand out as amoung the best systems for communications with the HP 3000. Of these systems the HP 150 is unique since it emulates the HP 2623 terminal supporting HPs' graphics products on the HP 3000. One feature which sets these systems apart from the others is that the terminal emulation code is in ROM. This allows these systems to be configured to boot up as a terminal without having to load any code into memory. I am disappointed in not being able to use the HP 150s' standard 256K of RAM as display memory.

Amoung the next group I would include the HP 200 series with the HP 2623 terminal emulation program and the IBM PC running PC2622. PC2622 is an execellent disc-based terminal emulation programs with integrated file transfer utility.

The HP 120/125 and the HP 86/87 both provide adequate terminal emulation and file transfer capabilities. Block Format/125 and Series 100/DSN/LINK lack a needed user friendly integration so that re-loading of code is not required when changing functions. Block Format could be significantly enhanced by emulating an HP 2622 instead of supporting VPLUS applications only, but for many users it will perform all or most of the needed functions. The HP 86/87 screen size is a limiting factor as are the applications available on this machine.

I would not consider VDTE, which functions to a limited degree, unless VT52 emulation and/or HP 2648 graphics capabilities are also desired. PCCOM failed nearly all of my block mode terminal emulation tests and obtaining help from DWJ associates proved to be very difficult and never prompt.

Alleyn-Day International
1721 Grove Street #3
Berkeley, California 94709
(415) 486-8202

Direct Incorporated
4201 Burton Drive
Santa Clara, CA 95054
(800) 538-8404/(408) 980-1414

DWJ Associates, Inc.
One Robinson Lane
Ridgewood, N.J. 07450
(201) 445-1711

Inner Loop Software
P.O. Box 45857
Los Angeles, CA 90045
(213) 822-2800

System Automation Corporation
8555 Sixteenth Street
Silver Spring, Maryland 20910
(301) 565-9400

Walker Richer & Quinn, Inc.
Lake Union Place, Suite 201
1914 N. 34th Street
Seattle, Washington 98103
(206) 634-0503

R. Gregory Stephens is a Computer Scientist at Lawrence Livermore National Laboratory working for Administrative Information Systems. He received his Bachelors of Science degree from California State University at Sacramento.

AN INTRODUCTION TO DATA BASE NORMALIZATION

Richard L. Seltzer
Prudential Reinsurance Company

PREFACE

The terminology of relational theory is not strictly adhered to in this paper. Wherever possible, IMAGE terminology is used in order to facilitate understanding by the reader. Consequently, this paper cannot be considered a formal exposition of normalization.

I would like to acknowledge the cooperation and support of Prudential Reinsurance during the preparation of this paper. Thanks also go to Dean Livingston, Linda Mitchell, and Zoe Putnam for invaluable critiques on various facets of this paper, and especially to Lisa Masarek for presenting the initial opportunity.

INTRODUCTION

Relational data bases have been getting much attention over the past several years. A major feature of relational data base theory is normalization of data. Hierarchical data bases and network data bases (of which IMAGE is a subset) can benefit from the techniques of data normalization in the conceptual design of the data structure, as one facet of entity relationship analysis. This paper will describe and illustrate the several forms of normalization, discuss their implementation in IMAGE data bases, and examine the tradeoffs of such an implementation.

The examples used in this paper are very simple and the illustrated result of normalization may not differ much from the way that most people would design a data base. The purpose of this paper is, however, to illustrate the rules of normalization so that in more complex cases, the approach to normalization would be understood and a methodology applied to data base design.

I. NORMALIZATION DEFINED

Normalization is a method of organizing the data in a data base and decomposing the data into small, simple, non-redundant related units which contain functionally related data. Six forms of normalization are discussed here1. They are first (1NF), second (2NF), third (3NF), Boyce/Codd (BCNF), fourth (4NF), and fifth (5NF) normal forms.2 The higher normal forms are more strongly decomposed than the lower normal forms and are inclusive of them.

There are several advantages that result from normalization of data that may be missed in unnormalized data structures. Many affect the data and the database itself:

* Data base storage requirements are kept to a minimum. The elimination of redundant data also eliminates the need for space that the redundant data would have required.
* Maintenance of data is both eased and speeded. When fields are updated, they need be updated in only one place, not in many, due to the lack of redundancy.
* Inconsistency of data is avoided. Because, in most cases, there is only one copy of data, multiple copies of data with differing values does not occur.

* Currency of data is assured. Since there are not multiple copies of inconsistent data, there is no question as to which copy is the most current one.

* Data base manipulation is facilitated. The functional grouping of like data in one place, rather than in several places, eliminates major restructuring of the data base when fields or data sets are added or deleted.

Another advantage is that the data become more useful. Ad hoc data structures may be good for the requirement at hand, but should new functions be required, these data structures can become quite cumbersome. Since normalized data is kept in small, functional blocks, there exists a flexibilty allowing for use in new and different functions.

The key to normalization's advantages, decomposition of data into functional groups, also results in its disadvantage: an application may have to perform more retrieval operations because the data required resides in several data sets.3

These advantages and disadvantages will be illustrated throughout this paper.

II. THE NORMALIZATION PROCESS

Before detailing the normalization process, two concepts need to be explained: keys and functional dependence.

A key is a value used to identify a record. (Do not confuse this key with the IMAGE search item, which is used for path implementation purposes.) A key may be one field, or it may consist of several fields, in which case it is known as a composite key. There are several types of keys.

A primary key is a value that uniquely identifies a record within a data set. Every record in a normalized data base must have a primary key. An example of a single field primary key is Part-no as the key in data set PARTS (which describes a particular part). A composite primary key would be Part-no and Warehouse-no as the key in data set PART-WAREHOUSE (which gives the quantity of a particular part at a particular location). A primary key value, or any part of it (as in a composite primary key), may not be null. If it were, it could not uniquely identify the record. Primary keys in the examples in this paper are denoted by underlining.

Candidate keys are multiple keys in a record, each of which uniquely identifies that record. For example, if each supplier has a unique name as well as a unique number, both Supplier-no and Supplier-name are candidate keys. Candidate keys in the examples in this paper are denoted by a pound sign (#).

A foreign key is a non-primary key value in a data set that is also a primary key in another data set. The individual values within a composite primary key are also considered foreign keys. A foreign key can be used to represent a relationship between data sets. For example, in a case where a particular shipper will be shipping a part on order from a certain supplier, the PARTS- SUPPLIER data set would contain a field for Shipper-no to indicate who will be shipping that part. All the information about that shipper would be found in the SHIPPERS data set under the primary key Shipper-no. Because they are needed to maintain relationships, foreign keys are the only redundant data found in a fully normalized data base. Foreign keys in the examples in this paper are denoted by an asterisk (*).

Functional dependence deals with the identification of the value of one field by another field or key. Field B is functionally dependent on field A if and only if for any A value there exists only one B value. For example, given Supplier-no (field A), you can determine the Supplier-address (field B); Supplier-address is functionally dependent on Supplier-no. Field A is also known as the determinant. Since the primary key uniquely identifies a record, the primary key is the determinant of all of the fields in that record.

The process of normalizing data will now be illustrated in a stepwise fashion.4 As an example, let's use a parts inventory system. The information necessary for this system, along with an explanation of the fields, is represented in Figure 1.

```
Part-no                 -- identifies a part
Part-cost               -- cost of the part
Parts-on-hand           -- quantity available of the part
Parts-on-hand-value     -- total inventory value of the part
```

```
Supplier-no-1          -- identifies a supplier of the part
Supplier-name-1        -- supplier's name
Supplier-address-1     -- supplier's address
Supplier-bal-due-1     -- total balance due the supplier for all orders
Parts-on-order-1       -- quantity of the part on order from the supplier
Shipper-no-1           -- identifies the shipper of the part on order
Shipper-name-1         -- shipper's name
Shipper-address-1      -- shipper's address
Supplier-no-2          -- identifies a second supplier of the part
Supplier-name-2        -- second supplier's name
Supplier-address-2     -- second supplier's address
Supplier-bal-due-2     -- total balance due second supplier for all orders
Parts-on-order-2       -- quantity of the part ordered from this supplier
Shipper-no-2           -- shipper of part on order from second supplier
Shipper-name-2         -- shipper's name
Shipper-address-2      -- shipper's address
Warehouse-no-1         -- identifies warehouse where part is stored
Warehouse-loc-1        -- location of the warehouse
Warehouse-capacity-1   -- capacity of the warehouse
Wh-parts-quantity-1    -- quantity of the part stored in the warehouse
Wh-manager-no-1        -- identifies the manager of the warehouse
Wh-manager-name-1      -- manager's name
Wh-manager-salary-1    -- manager's salary
Warehouse-no-2         -- another warehouse where the part is stored
Warehouse-loc-2        -- location of second warehouse
Warehouse-capacity-2   -- capacity of second warehouse
Wh-parts-quantity-2    -- quantity of the part stored in second warehouse
Wh-manager-no-2        -- identifies manager of the second warehouse
Wh-manager-name-2      -- manager's name
Wh-manager-salary-2    -- manager's salary
```

FIGURE 1
Unnormalized Data

This data organization is prone to wasted space. For example, a part can be supplied by many suppliers. If a particular part is supplied by only one supplier, the information relevant to that supplier will be put in Supplier-no-1, Supplier-name-1, Supplier-address-1, and Supplier-bal-due-1. The fields for Supplier-2 (and however many more suppliers there are) will be empty.5

First normal form resolves this problem by removing the repetitious fields. A data set is in 1NF if and only if it has no repeating fields. The use of IMAGE sub-items would violate 1NF.

In order to place our data into 1NF, the Supplier, Shipper, Warehouse and other repititious fields must be replaced by single sets of fields. The data after this transformation, now in 1NF, appear in Figure 2.

Each record in this data set, which has been labeled GENERAL-PART-INFO, can be uniquely identified (i.e., determined) by a composite key made up of Part-no, Supplier-no, and Warehouse-no. Every field is filled6, and there is no wasted space.

```
GENERAL-PART-INFO

Part-no
Part-cost
Parts-on-hand
Parts-on-hand-value
Supplier-no
Supplier-name
```

```
Supplier-address
Supplier-bal-due
Parts-on-order
Shipper-no
Shipper-name
Shipper-address
Warehouse-no
Warehouse-loc
Warehouse-capacity
Wh-parts-quantity
Wh-manager-no
Wh-manager-name
Wh-manager-salary
```

FIGURE 2
Data in 1NF

There are still problems with this form, though. For example, if a supplier is not currently supplying a part, or a warehouse is not currently being used, information about that supplier or warehouse will not be in the data base until a part is supplied by that supplier or a part is stored in that warehouse. Similarly, if only one part is supplied by a particular supplier, and it is no longer purchased from him, the information about that supplier is lost. Keeping the supplier or warehouse information in a record with a null Part-no would not be a solution; that would violate the rule that no value in a primary key may be null.

Another problem arises with the information that is in this data set. It is redundant. If a supplier supplies three different parts, information about that supplier, such as the address, appears three times. This creates extra work for updating. Whenever the address changes, every record in which it appears must be updated. If not every record is changed, the data base becomes inconsistent, leading to other troubles, such as determining which datum is the most current.

The logical solution to these problems, and the idea behind second normal form, is to somehow extract this redundant information about a specific entity and deposit a single copy of it somewhere else. This is accomplished by creating data sets containing information which is dependent on the entire primary key and not on only part of the (composite) primary key. (This type of functional dependence is called full functional dependence.) For example, the composite key Part-no Warehouse-no determines the quantity of a particular part in a particular warehouse. A field such as part-cost is functionally dependent only on Part-no and not on Warehouse-no and is thereby not fully functionally dependent on the entire key Part-no Warehouse-no.

A data set is in 2NF if and only if it is in 1NF and every nonkey field is fully dependent on the primary key.

To decompose GENERAL-PART-INFO into 2NF, the fields that are only partially dependent on the primary key will be removed to separate data sets with a primary key on which they are fully dependent. The original primary key does remain in the original data set, and its components function as foreign keys. In our example, six new data sets will be created, as appear in Figure 3, and all of the fields in them are fully functionally dependent on their respective primary keys. The data set GENERAL-PART-INFO, renamed PART-SUPPLIER-WAREHOUSE after the decomposition, denotes the relationship among parts, suppliers of those parts, and warehouses in which those parts are supplied.

```
PARTS                 SUPPLIERS             WAREHOUSES

Part-no               Supplier-no           Warehouse-no
Part-cost             Supplier-name#        Warehouse-loc
Parts-on-hand         Supplier-address      Warehouse-capacity
Parts-on-hand-value   Supplier-bal-due      Wh-manager-no
                                            Wh-manager-name
                                            Wh-manager-salary
```

PART-WAREHOUSE	PART-SUPPLIER	PART-SUPPLIER-WAREHOUSE
Part-no*	Part-no*	Part-no*
Warehouse-no*	Supplier-no*	Supplier-no*
Wh-parts-quantity	Parts-on-order	Warehouse-no*
	Shipper-no	
	Shipper-name	
	Shipper-address	

FIGURE 3
Data in 2NF

Even though there are now six data sets instead of one, the total information in them is preserved through relationships expressed by the foreign keys. For example, from the data set PART-WAREHOUSE, which contains the quantity of a particular part stored in a particular warehouse, information pertaining uniquely to the part can be obtained by taking Part-no and finding the record in the PARTS data set containing the same Part-no. The same thing can be done with Warehouse-no and the WAREHOUSE data set.7

Despite the fact that the data sets are in 2NF, there are still some redundancy problems. For example, a shipper may have information about him recorded in several places if he is shipping several parts, leading to data consistency and currency questions; information about the shipper will be lost if he is not shipping any parts at this time.

These problems arise because the data in question is information not about the primary key, but about another non-key field. The data does not directly describe (or depend on) the primary key, but does so transitively through the non-key field. Take as an example the data set WAREHOUSES. Wh-manager-salary is an attribute of Wh-manager-no (the manager earns a specific salary) and is directly dependent on it. Wh-manager-no is an attribute of Warehouse-no (the warehouse has one manager) and is directly dependent on it. Consequently, for a warehouse, there is only one manager-salary paid, and Wh-manager-salary is therefore transitively dependent on Warehouse-no by way of Wh-manager-no.

The solution to the redundancy problems is to extract this transitively dependent information and place it in its own data set. This is accomplished by third normal form.

A data set is in 3NF if and only if it is in 2NF and every nonkey field is directly dependent on the primary key.

Decomposition of 2NF data sets into 3NF data sets is done by removing to their own data set the non-key data that directly describe other non-key data. The primary key of the new data set is the "directly-described" non-key data. A copy of this key remains in the original data set to serve as a foreign key.

To decompose the data in Figure 3 into 3NF, as represented in Figure 4, three new data sets, SHIPPERS, MANAGERS, and PART-VALUES8, are created. (Note that the fields Part-cost and Parts-on-hand form a composite primary key in the data set PART-VALUES. Therefore, these two fields together form a composite foreign key in the data set PARTS.)

(3NF does have a weakness in that it does not satisfactorily handle cases of two or more composite and overlapping keys.9 As a result of this shortcoming, 3NF has been replaced by Boyce/Codd normal form, which is stronger yet conceptually simpler. A data set is in BCNF if and only if every determinant is a candidate key. This means that every field in the data set must be directly functionally dependent on the entire primary key, and only candidates of the entire primary key are allowed. As you can see, Figure 4 is also in BCNF.10)

PARTS	SUPPLIERS	WAREHOUSES
Part-no	Supplier-no	Warehouse-no
Part-cost*	Supplier-name#	Warehouse-loc
Parts-on-hand*	Supplier-address	Warehouse-capacity
	Supplier-bal-due	Wh-manager-no*

PART-VALUES	PART-WAREHOUSE	PART-SUPPLIER
Parts-on-hand	Part-no*	Part-no*
Part-cost	Warehouse-no*	Supplier-no*
Parts-on-hand-value	Wh-parts-quantity	Parts-on-order
		Shipper-no*

SHIPPERS	MANAGERS	PART-SUPPLIER-WAREHOUSE
Shipper-no	Manager-no	Part-no*
Shipper-name#	Manager-name#	Supplier-no*
Shipper-address	Manager-salary	Warehouse-no*

center 2 FIGURE 4 Data in 3NF/BCNF

Fourth normal form deals with multivalued dependencies. A multivalued dependence (MVD) can be considered a special case of functional dependence, but instead of one specific field value associated with a given key, there is a specific set of values for that key.

Putting two or more MVD's that are independent of each other (i.e., they are dependent on the primary key, but have no effect on each other) in the same data set leads to redundancy problems. For example, a shipper can make deliveries in one or more time periods -- day, night, or weekend. The shipper can also handle, independent of delivery time, one or more kinds of freight -- regular, fragile, electronic, or perishable -- and have a rating for each type. One way of arranging this data is as in Figure 5. This data set is in BCNF -- the entire record is the primary key. Still, redundancy problems remain. Add another type of freight for a shipper, and many records, not just one, will have to be added to the data set. Updating of records will also suffer from the effects of data redundancy. Independent MVD's will have to be put in separate data sets to avoid these problems.

Shipper-no	Delivery-time	Freight-type
1	D	R
1	D	F
1	D	E
1	D	P
1	N	R
1	N	F
1	N	E
1	N	P
1	W	R

1	W	F
1	W	E
1	W	P
2	D	R
2	D	F
2	D	P
2	N	R
2	N	P
2	W	R
2	W	P
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
143	D	R
143	D	F
143	N	R
143	N	F
143	W	R

FIGURE 5
MVD's in BCNF

A data set is in 4NF if and only if it does not contain more than one independent MVD about an entity.

To decompose a data set into 4NF, each MVD is paired with the primary key it is dependent upon to form a new composite primary key for a new data set. Any information that is fully and directly dependent on this new primary key is removed from the original data set and placed in the new data set.

If, in our data base, we want to keep information about a shipper's delivery times, the types of freight he carries, and, additionally, his rating for that type of freight, the data will be decomposed into two small data sets11:

Shipper-no Delivery-time &
Shipper-no Freight-type Rating

The full data base in 4NF, with these new data sets, appears in Figure 6.

PARTS	SUPPLIERS	WAREHOUSES
Part-no	Supplier-no	Warehouse-no
Part-cost	Supplier-name#	Warehouse-loc

Parts-on-hand	Supplier-address Supplier-bal-due	Warehouse-capacity Wh-manager-no*
PART-WAREHOUSE	PART-SUPPLIER	PART-SUPPLIER-WAREHOUSE
Part-no* Warehouse-no* Wh-parts-quantity	Part-no* Supplier-no* Parts-on-order Shipper-no*	Part-no* Supplier-no* Warehouse-no*
SHIPPERS	MANAGERS	
Shipper-no Shipper-name# Shipper-address	Manager-no Manager-name# Manager-salary	
SHIPPER-TIMES	SHIPPER-FREIGHT	
Shipper-no* Delivery-time	Shipper-no* Freight-type Rating	

FIGURE 6
Data in 4NF

Fifth normal form provides a general purpose decomposition. It is conceptually based not only on the determination of fields in the data set, as are the other normal forms, but also on the values within the data set. Consequently, it can be difficult to explain 5NF and to identify 5NF data sets. Therefore, what follows is only a brief overview.

5NF, like 4NF, deals with multiple MVD's, but in cases where they are not independent of each other. Take the PART-SUPPLIER-WAREHOUSE data set created as a result of 2NF. It contains MVD's, since a part could come from many suppliers and could be stored in many warehouses. These MVD's are not, however, independent of each other. All of a supplier's parts are not necessarily in all of the warehouses.

A data set is in 5NF if and only if it cannot be further decomposed without causing the creation of spurious information upon reconstruction.

For a non-5NF data set to be decomposed into 5NF, it must meet a certain constraint. Using our example, it can be stated like this: if a part

is stored in certain warehouses, and that part is obtained from certain suppliers, then the part from each and every supplier will be in each and every warehouse.12

When this constraint holds, a data set with dependent MVD's can be decomposed into three or more (not two, as in the previously mentioned normal forms) smaller data sets, from which the original information can be accurately reconstructed by combining all of these data sets if the original data set meets the specific constraint. However, when the constraint does not hold, a recombination of the smaller data sets results in extra, spurious information.

Figure 7 illustrates a case where the data set cannot be decomposed and accurately recreated. PART-SUPPLIER-WAREHOUSE is decomposed into three smaller data sets PART-SUPPLIER, SUPPLIER-WAREHOUSE, and PART-WAREHOUSE (the PART-WAREHOUSE data set already existing in our data base serves this purpose). When PART-SUPPLIER and SUPPLIER-WAREHOUSE are joined by the common value of their common key, Part-no, and the result joined with PART-WAREHOUSE by the common values of their common keys Part-no and Warehouse-no, the reconstructed data set contains an extra, spurious record (P1, S1, W1). (Spurious records are indicated in the example by an exclamation point (!).) Therefore, the original PART-SUPPLIER-WAREHOUSE data set is in 5NF.

If the record (P1, S1, W1) were in the PART-SUPPLIER-WAREHOUSE data set, it could be accurately reconstructed from the decomposed data sets. In that case, PART-SUPPLIER-WAREHOUSE would not be in 5NF, but would have to be decomposed into the three smaller 5NF data sets.

ORIGINAL DATA SET

Part-no	Supplier-no	Warehouse-no
P1	S1	W2
P1	S2	W1
P2	S1	W1

DECOMPOSED DATA SETS

Part-no	Supplier-no
P1	S1
P1	S2
P2	S1

Supplier-no	Warehouse-no
S1	W1
S1	W2
S2	W1

Part-no	Warehouse-no
P1	W1
P1	W2
P2	W1

RECOMPOSITION (INTERMEDIATE STEP)

	Part-no	Supplier-no	Warehouse-no
!	P1	S1	W1
	P1	S1	W2
	P1	S2	W1
	P2	S1	W1
!	P2	S1	W2

Part-no	Warehouse-no
P1	W1
P1	W2
P2	W1

RECOMPOSED DATA SET WITH SPURIOUS RECORD

Part-no	Supplier-no	Warehouse-no

!	P1	S1	W1
	P1	S1	W2
	P1	S2	W1
	P2	S1	W1

FIGURE 7
Dependent MVD Decomposition and Recomposition

III. IMAGE IMPLEMENTATION

We now have our data base decomposed into several small, functionally cohesive, nonredundant data sets. A logical view of these data sets and how they relate to each other is presented in Figure 8. The arrows represent whether a one-to-one, one-to-many, or many-to-one relationship exists between data sets.

As you can see, there are several levels of dependence. IMAGE, however, being only a two level network data base management system, can efficiently handle only one dependence level for one-to-many relationships, making hierarchy implementation awkward.[13] It also has, like any data base management system, implementation features to consider. How do we bring the logical data base into conformity with the real world of IMAGE? Here are some general rules (of course, general rules always have exceptions):

1. Data sets that have a single-field primary key, such as the data set PARTS, should be manual master data sets. This way, a search for a value, for example, that of a foreign key in another data set, can hash directly to the proper record.

2. Data sets that have a composite primary key, such as the data set PART- SUPPLIER, should be detail data sets. A composite key implies a one-to- many relationship between master data sets having single-field keys and the composite-key detail data set itself; consequently, paths between these data sets can be established. Automatic master data sets can be established to create paths for key fields without corresponding master data sets (for example, an automatic master for FREIGHT with a path to SHIPPER-FREIGHT) or for non-key fields for which quick access is a necessity.

3. Those manual master data sets requiring quick access of non-key fields can be dropped a level to detail data sets, and automatic master data sets can be created to establish the search paths. (I must state here the standard warnings about many paths in a detail data set: while retrieval is fast, updating is slow, and maintenance of the data base can be difficult.)

The IMAGE-implemented normalized data base is shown in Figure 9. Standard HP graphic representation is used.

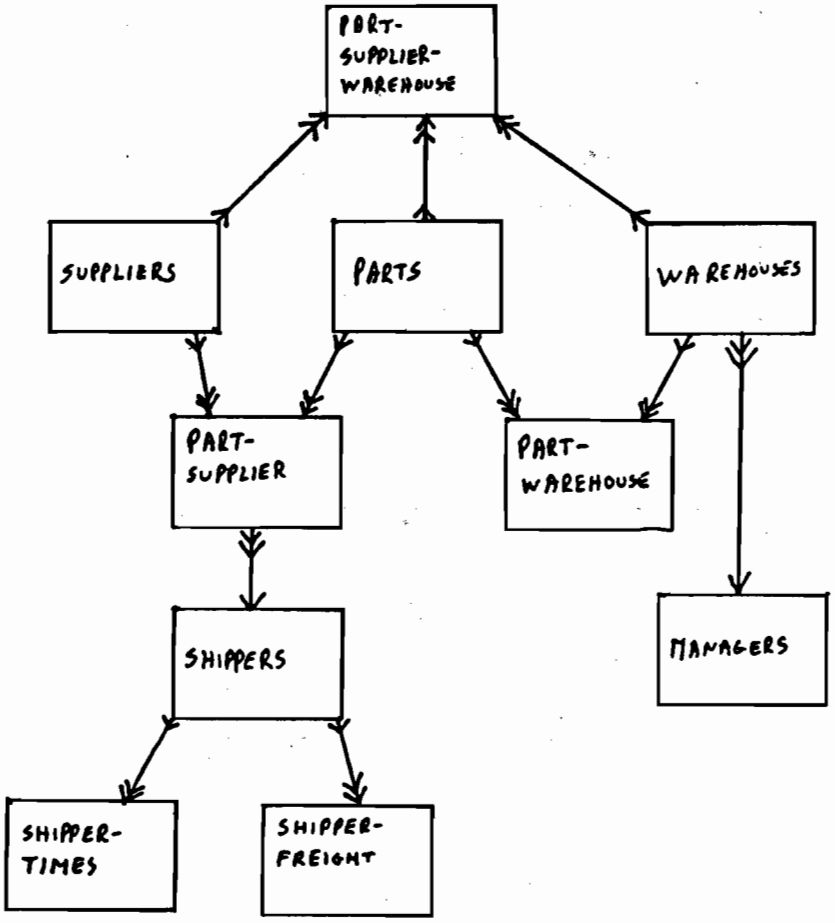


Figure 8
Logical Relationships Among the Data Sets

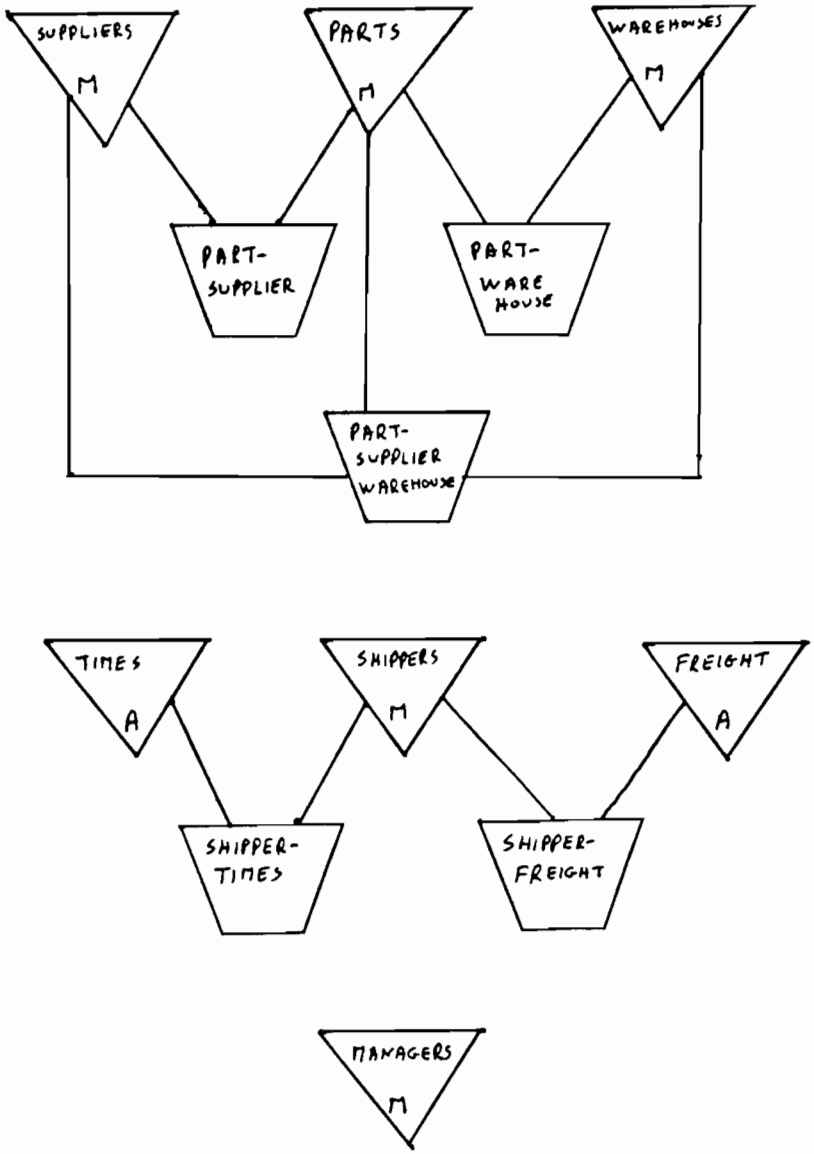


Figure 9
IMAGE Implementation of Data Base

Figure 9
IMAGE Implementation of Data Base

CONCLUSION

Normalization of data provides many advantages. The data base requires much less space than an unnormalized data base would. Because of the nonredundancy of data, the data base becomes more useful and can be maintained much more easily, both physically (except for IMAGE search items, which are often the redundant keys) and logically (as shown in the examples).

On the other hand, application programs must retrieve data from many data sets instead of from only one, thereby increasing retrieval time. The question then arises, how many levels of normalization to apply to the data before the cost of extended retrieval time overshadows the benefits of normalization. The answer is application dependent, but it would be difficult to justify not commonly normalizing to at least 3NF/BCNF.

NOTES

1. There are other normal forms, such as (3,3) normal form and domain-key normal form. These, however, are beyond the scope of this paper.

2. 1NF, 2NF, and 3NF were discovered by E. F. Codd (see Codd[1970] and Codd[1972]). BCNF was discovered by Codd and R. F. Boyce (see E. F. Codd, "Recent Investigations into Relational Database Systems." Proc. IFIP Congress 1974.). 4NF ad 5NF were discovered by R. Fagin (see R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Databases." ACM Transactions on Database Systems. 2, No. 3 (September 1977). and R. Fagin, "Normal Forms and Relational Database Operators." ACM SIGMOD International Conference on Management of Data.).

3. To make the data base structure transparent to the applications, shield the applications from data base maintenance, and prevent excessive program maintenance, an access module might be used. When the structure of the data changes, only the access module needs to be changed, not the application programs. The access module itself would add slightly to retrieval time.

4. It is interesting to compare stepwise normalization of data into nonredundant functional groups with structured programming's idea of stepwise refinement of programs, as presented by Wirth and Dijkstra, and nonredundant functional procedures/modules. One could be called "structured data" and the other "'normalized' programs" (although there is no formal method of which I know for "normalizing" programs). (See Wirth, Systematic Programming (1973), Algorithms + Data Structures = Programs (1975) and Dijkstra Structured Programming (1972).)

5. This type of data organization resulted in a real-life problem on an IMAGE data base. A detail data set had a series of names, Name-1, Name-2, Name- 3, and Name-4, which were dependent upon the record key. All four names were not always entered. Sometimes only one name was entered, in Name-1; sometimes only two names were entered, in Name-1 and Name-2; and so on. Most of the time, no Name-4 was entered. Those name fields not entered were filled, by default, with blanks.

   These name fields were often used for searching this data set, so they were made search items and linked to an automatic master. Unfortunately, because of all the blank entries in Name-4, the maximum chain length was quickly reached for the blank values and no more records could be added to that data set unless all the name fields were filled.

6. That is, there is no systematic occurrence of sets of empty fields. Only fields that are part of a key are forbidden to be null. Non-key fields may contain null values, but these should be rarely occur.

7. When physically or logically reconstructing the data through the foreign keys, multiple combinations of the reconstructed records become possible. For example, given the data sets:

PARTS WAREHOUSES PART-WAREHOUSE

Part-no	Part-info
PN1	PI1...
PN2	PI2...

Warehouse-no	Warehouse-info
WN1	WI1...
WN2	WI2...
WN3	WI3...

Part-no	Warehouse-no
PN1	WN1
PN1	WN2
PN1	WN3
PN2	WN1
PN2	WN2
PN2	WN3

the following redundant data set would result from recombination:

PART-WAREHOUSE-COMBINED

Part-no	Warehouse-no	Part-info	Warehouse-info
PN1	WN1	PI1...	WI1...
PN1	WN2	PI1...	WI2...
PN1	WN3	PI1...	WI3...
PN2	WN1	PI2...	WI1...
PN2	WN2	PI2...	WI2...
PN2	WN3	PI2...	WI3...

Consequently, we see that the amount of space saved by decomposition is a multiplicative function. This concept is highlighted by the fact that the relational operator used for reconstructing data sets through common keys, join, is based on the traditional set operator Cartesian product.

8. PART-VALUES appears here for illustrative purposes. While it is in 3NF, it is really nothing more than a multiplication table. Since the Parts- on-hand-value field can be calculated, this data set is actually unnecessary and its presence contradicts the philosophy behind normalization. For this reason, the PART-VALUES data set will not appear subsequently in examples. Pragmatically, the relative importance of processing time to storage space is the deciding factor in whether data that can be calculated should be included in a data base.

9. As an example, take the following case:

Part-no Supplier-no Supplier-name Parts-on-order

Because Supplier-no and Supplier-name are candidate keys, this data set has two overlapping candidate primary keys: Part-no Supplier-no, and Part- no Supplier-name. This data set is in 3NF, since every non-key field is directly dependent on the primary key. However, there is still a possibility of redundancy in Supplier-name. This is because 3NF does not require a field to be fully dependent on the primary key if it is itself a component of a candidate key.

10. BCNF can sometimes be too strong, destroying the dependence relationships of the decomposed data sets. As an example, take the data set

Project Leader Salary

This data set is in 2NF but not 3NF nor BCNF. It is possible to decompose this data as such:

Project Leader & Project Salary

These two data sets are in BCNF, but the dependence relationship and the independence of the data have been lost, resulting in insertion, deletion, and update irregularities. For example, it is not possible to know a given project leader's salary unless that project leader is manging a project, and if he is managing two projects, his salary is recorded twice.

It is also possible to decompose the data this way:

Project Salary & Leader Salary

These, too, are in BCNF, but if we combine these two data sets (logically or physically) based on the common field Salary, extra, spurious records will be created. This results from the fact that many project leaders may earn the same salary, and the combined data set is a product of all possible combinations of the smaller data sets linked by common salaries.

The optimal decomposition, reflecting the hierarchy of the relationships, is this:

Project Leader* & Leader Salary

This structure shows which project leader manages a given project, and we know a project leader's salary even when he is not leading any project.

11. There are perfectly valid reasons for having one data set with all of the information, such as when you need information that is fully dependent on several multivalued dependencies (this is really a result of 2NF). For example, to record the on-time percentage of a given shipper at a given time for a specific type of freight, the following data set is needed:

Shipper Delivery-time Freight-type On-time-percentage

From the information in this data set, you can calculate the on-time- percentage for a single delivery-time or a single freight-type, or for any combination. If you don't want to calculate this information, you can build new data sets (e.g., Delivery-time Freight-type On-time- percentage) to house that information, but that saves calculation time at the expense of extra space consumption and data redundancy.

Note that even in this case, the data set

Shipper-no Freight-type Rating

will still exist to prevent redundancy within rating type. Of course, On-time-percentage could be a part of this data set, too, under the "save time, not space" option.

12. More formally, the constraint, called a join dependence, can be described this way:

If the records

(P1, S1, W2)
(P1, S2, W1)
(P2, S1, W1)

appear in the data set PART-SUPPLIER-WAREHOUSE,
then the record (P1, S1, W1)
appears in the data set PART-SUPPLIER-WAREHOUSE.

13. For example, if data set A has a one-to-many relationship to data set B, and data set B has a one-to-many relationship to data set C (not illus- trated in this paper), in the best case, retrieval of a record from data set C requires a chain traversal from master data set A to detail data set B, then a hash to master data set C', from which another chain traversal to detail data set C is done. One-to-one and many-to-one relationships are easily handled by hashing directly to the entry in a master data set.

BIBLIOGRAPHY

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM. 13, No. 6 (June 1970) 377-387.

---------. "Further Normalization of the Data Base Relational Model." Data Base Systems (Courant Computer Science Symposium 6). Ed. Randall Rustin. Englewood Cliffs, N.J.: Prentice-Hall, 1972. 33-64.

Date, C.J. An Introduction to Data Base Systems. Third Edition. Reading, Mass.: Addison-Wesley, 1981.

Flores, Ivan. Data Base Architecture. New York, N.Y.: Van Nostrand Reinhold, 1981.

IMAGE Data Base Management System Reference Manual. Second Edition. Cupertino, Ca.: Hewlett-Packard, 1979.

Kent, William. "A Simple Guide to Five Normal Forms in Relational Data Base Theory." Communications of the ACM. 26, No.2 (February 1983) 120-125.

Ullman, J.D. Principles of Database Systems. Second Edition. Rockville, Maryland: Computer Science Press, 1982.

Further references may be obtained from the excellent bibliographic notes in Date.

Richard Seltzer is familiar with a variety of hardware, having his most extensive experience on HP 3000's at the Prudential Reinsurance Co. in Newark, N.J. Among

his areas of involvement on the HP 3000 are data base design, the design and programming of both business and software systems (such as security and chargeback systems), and systems management. He is currently pursuing an M.S. in Computer Methodology from Baruch College of the City University of New York. For his thesis, he is writing a structured English query language based on relational algebra.

ALLOCATING DATA CENTER COSTS

By Gary Leight
Project Resources, Inc.

Once upon a time data processing was a function of a company's accounting department. Computers were fancy tools for bookkeeping, and little more. And the cost of running these machines was considered, quite rightly, to be part of the accounting budget.

As the value of computers became better understood, their use soon spread to other areas of the company. And the expenses of data processing spread as well. Allocating costs was simple in those days. Logs were kept by hand, and the total DP bill was divided by the number of hours of use, measured by a wall clock. Each user was charged a prorated share. This was the first data processing chargeback system.

With the advent of multiprogramming, things became more complicated. With jobs running concurrently, usage logs could no longer be kept by hand. More sophisticated methods had to be developed, so that the computer could monitor and record its own use. The modern chargeback system was born.

A company can view data processing costs as charges against the individual user departments, or as part of its total overhead. However, to coin a business adage, if you get enough overhead, the sky usually falls. By allocating DP costs to the user, the corporate management gains some measure of control over data processing expenditures. As information processing becomes a part of most corporate functions, failure to include the costs of processing in user departments' profit and loss statements can be a financial distortion. Management risks coming to wrong conclusions based on false cost levels if an operation uses central DP services not contained in its total budget. An effective cost allocation system should prevent this.

In addition to keeping management informed of the resources being used, a chargeback system serves to discourage unnecessary usage. If users are made to think of resources in terms of money, they are far less likely to be wasteful. Of course this principle applies to telephone calls and paper clips as well as to computer usage. The question is simply, which resource costs does management wish to bring to the user's attention? No businessman wants to quibble about paper clips at the risk of alienating the workforce, but on the other hand, there are obvious financial advantages to resource conservatism. It's likely we can all agree that DP time is sufficiently expensive to be kept track of.

The basic tenets of economics can be applied to managing the demand for resource use. By placing a high price on one resource relative to another (for example, prime-shift versus nighttime processing), the organization can alter user demand for a particular resource, and create a better balance in the use of available capacity. Consider how effectively the phone company encourages usage after business hours by offering discount rates. At times it may be best for the organization to discontinue the availability of a certain resource, but generally a high enough price will lead users to discover alternatives.

DP costs can be allocated to achieve either full or partial recovery. In a full-recovery approach, the object is to zero out the cost of the DP center, so that every dollar of expense is assigned to DP users. The easiest way to achieve this is to identify the services, units of work, resources, and other items for which a charge is to be made, and to treat them as a product line. Rates for each resource or service are determined by dividing estimated total cost by estimated use. In theory, this method of rate setting results in full recovery of costs. In practice, however, neither the budget forecast

of costs nor the estimate of resource use will ever be exact. The company can either make an after the fact adjustment, or absorb the amount unallocated into the corporate overhead. The more care taken with the budget forecasts, the less variation there will be from a perfect zero balance.

In a partial-recovery approach certain services and resources are charged out and others are treated as overhead. Decisions on which resources to charge for will vary with different management philosophies.

Since the objective is cost recovery, the budget or expenditure plan for the year must be prepared so that anticipated costs are identified in advance.The DP department can prepare a single budget covering all functions, but the chargeback scheme can be more easily developed if a separate budget is prepared for each functional area. One of the advantages to a successful chargeback system is that it is self-perpetuating. The data it generates serves to correct mistakes in the previous budget assessment, thus guaranteeing more accurate budgets in the future.

If a chargeback system is based on charging for use of resources at a unit rate, achieving dollar target objectives depends on accurately predicting use. Either of two bases, anticipated actual use or maximum possible use, can be employed to estimate use levels.

The philosophy of setting rates based on anticipated actual use is to have each resource fully recover its costs on the basis of whatever use is made. This means that significant shifts in use require rate adjustments to avoid recovering too much or too little. This approach makes users' costs sensitive to resource utilization by other users. This, in turn, makes users sensitive in general. For example, implementing a major new system will reduce unit rates, and therefore, current users' costs, since utilization increases while costs to be recovered remain relatively fixed. However, if a user drops out, those remaining must each shoulder a greater portion of the total cost.

When setting rates on the basis of maximum possible use, the cost of excess capacity is absorbed internally. Although use levels change, rates remain the same since they are based on the theoretical maximum achievable use level for each resource measured. This stability of rates is generally preferred by users over the previous method. If the organization does not object to unallocated costs for excess capacity, this method is the better of the two.

In describing various cost allocation systems, much space is usually given to the selection of which resources to measure. The answer to this is relatively simple. Measure everything that it is cost-efficient to measure and that can provide better understanding and control of the utilization of the overall system. Provided that the information is presented in a suitable fashion, complete information makes the entire realm of DP usage easier to visualize and to manage.

A comprehensive cost allocation and chargeback system for the HP3000 should at a minimum measure:

```
Batch and session usage
Daily, weekly and monthly disc space usage
Prime and non-prime time usage
Port connect time
Batch wall time
CPU seconds
Special paper or form usage
Line printer usage
```

Detail and summary reports should be available by job, user, account, application, department, port, printer, and time period. An effective cost allocation system should provide management with precise and complete auditing and accounting data.

A more sophisticated chargeback system would also allow sliding cost schedules for different jobs and sessions. An infinite number of these time/price schedules could be made available to the various corporate departments. By allocating percentages of job/session costs among user departments, the system could gain accuracy without enforcing a complex set of account structure requirements. The ability to roll up usage statistics and chargeback costs to a particular application would add a further degree of sophistication.

The efficient chargeback system is a tool designed by programmers for the benifit of management. But it is important in implementing a chargeback system, to keep the

psychology of the user in mind. The phrase "user friendly" is grossly overworked in software descriptions What is called for here is a system which appears to be "user friendly" but is in fact "user intimidating". Cost allocation reports to the user only serve to curb waste to the extent that they instill a sense of guilt. Thus, informing a non-technical user that he has spent 1.46 CPU hours is less effective than billing him for $262.80 admittedly fictional dollars.

Cost allocation and chargeback software is a practical business investment, and as such will pay for itself. The ideal system provides your organization with accurate and comprehensive data for job accounting, performance evaluation, billing and auditing. It makes session and batch analysis available to monitor productivity, system utilization, and performance. It enables the data center to identify service, turnaround, and cost information for all user areas. It helps the user to increase their own efficiency, and it informs management of how the data center budget is being spent.

CONCLUSION

To understand what modern chargeback software can do for your company, simply think of all data processing activities as a large business lunch. As the various departments and users bicker over who had the cottage cheese salad, the chargeback system is the exasperated waiter who patiently writes up seperate checks.

Gary Leight, President of Project Resources

Gary Leight is responsible for the Business Management functions of Project Resources, Inc. During his four years with the company, he has been responsible for:

Expanding the company from three to over a twenty data processing professionals

Broadening the companies product/services offering to include OCS, The Operations Control System; PBS, The Professional's Billing System and The Guardian Access Control System, in addition to time- sharing, consulting, programming, design and training.

For two years, Mr. Leight was the Western Regional Sales Manager for BTI Computer Systems.

For three years, Mr. Leight was a Commercial Sales Representative for Hewlett-Packard Company where he was responsible for penetrating 5 major corporate accounts and selling over 25 HP3000/2000/1000 computer systems.

For two years as a Systems Engineer with Hewlett-Packard, Mr. Leight was responsible for the sales and installation support of computer sys- tems in the Los Angeles area.

As a Programmer/Analyst for Raytheon Company in Wayland, Massachusetts, Mr. Leight participated in the design and development of a missile tracking and control system.

Mr. Leight has a BSEE and an MSEE from Rensselaer Polytechnic Institute in New York and an MBA from Santa Clara University. He has also attended over 1000 hours of training courses given by a wide range of technical, sales and management training organizations.

Process Handling for Fun and Profit

Jeff Kell
UTC Computing Services

Process Handling Fundamentals

A program with process handling capability may create, delete, and exert certain controls over other processes. Processes created by a program of this type are called son processes, the originator of the processes is called the father process. This relationship may extend to other 'generations' of processes as well.

In essence, a process handling program has the capability of doing a ':RUN' command. In the most simple form, a program creates some other program, suspends itself, and the newly created program begins execution. When the new program completes, the original program is reactivated. However, the father program does not have to suspend. It may continue executing some other task, or create additional son processes.

This concept can be applied to improve on many application systems, as will be explored later, but it also adds a degree of complexity to the operation. Most process handling applications can be duplicated, at least in terms of results, by standard programming techniques; but process handling can improve throughput, performance, and flexibility of certain applications. Thus, a tradeoff point is evident when the advantages of process handling cannot justify the extra code required to support reliable process handling applications.

Process Handling Difficulties

The most prominent difficulty in any process handling application is the need for error detection and handling, especially when more than one process is created, or the depth of process handling extends beyond the initial generation. Whenever the father process continues execution in parallel with the son processes, it is difficult enough to simply detect the completion of the son, let alone detect errors such as program abort conditions.

When only 1 son process is involved, and the father suspends while the son executes, the father will always be activated upon completion of the son process (or abnormal termination). Simple communication mechanisms such as a JCW may be used to determine the termination status of the son. When multiple son processes are involved, there is always the possibility of a son terminating while the father is already activated, in which case the activation has no affect; thus the father is unaware of the son's termination. There is no overall method of reliably detecting this condition. To overcome this situation, some form of periodic activation of the father can be performed so that the father can check on the status of each son.

Communication is often necessary between the members of a process handling family, if only to communicate status information. There are numerous ways of passing information, such as PARM, INFO, mail, message files, and data segments. The latter three allow for two way communication, and are usually the tools of choice. For example, a father process may have a message file for status reception. When a son terminates, the termination status and son identification are written to the message file. The father process can then read the message file until all sons have completed. If the father process uses 'timeout' reads on this file, which is possible, it can check for any abnormally terminated sons on each cycle.

When many processes are involved, the number of required message files may become too large

to be considered efficient. This becomes readily apparent when two way communication is required, as each son process must have its own message file. In such cases, a shared data segment is the most efficient communication mechanism. If this route is chosen, special procedures must also be written to support the transfer of messages through the data segment.

Theoretical Divisions of Process Handling

Process handling applications fall into one of two broad categories. The first category, independent processes, encompasses most typical uses of process handling. Independent processes do not depend upon other processes for their welfare. The second category, dependent processes, is much more complex. These processes work together to accomplish a common goal, and usually involve a high degree of communication and control.

Some applications do not fall into a specific division, and may exhibit characteristics of both. Independent processes have only a limited value in terms of processing advantages, but dependent processes are worthy of greater investigation. Applications which combine both areas are useful if the goal is to control several independent tasks, with the tasks being accomplished through dependent processes.

The most common form of independent process control is the menu driver. A menu driver program displays a set of command names and allows the user to select a command from the list by name, function key, or some other method. The menu driver, in turn, executes the program corresponding to that function. Menu drivers are often added to systems which are comprised of several separate programs to ease the transition from one program to another. The menu program in this case is suspended while the selected program executes.

Slight revisions to this facility could allow the user to select several "background" tasks to be executed while an on-line function is being done. For example, several reports may be requested while the user executes an on-line update program.

Menu programs often provide control information to the selected programs. For example, the menu program may set up file equations, obtain various run options, etc. In this way, extended control is provided over the selected programs.

The impact of such applications on the overall performance of the system varies on the implementation. In normal execution, MPE has a CI process for each user. When a :RUN is requested, the program is added as a son process of the CI. Thus, 2 processes are present for each user on the system. Thus, given a number of users (U), a number of user programs (P), and a fixed number of processes required by MPE (S), we can examine the number of processes required to support a given implementation of application systems. Note that other resources (data segments, etc.) also increase in proportion to the number of processes.

Without process handling, the number of processes is:

(a) U + P + S, or 2P + S

This equation holds for batch as well as on-line. Each additional user will require 2 processes. With a given physical limit of 256 processes under MPE, the limits on the number of users on a Series 64 (without MPE-V) are clearly defined by (256-S)/2.

Simple menu drivers (with a single son task) clearly complicate the matter, as both a CI process and a menu process will be required for each user. For this case, the number of processes is:

(b) 2U + P + S, or 3P + S

Menu drivers which allow multiple processes help somewhat by removing the CI process for each additional task, but the tasks must be batch oriented since the terminal user can only run one on-line task at a time. This equation is given by:

(c) 2U + P + S, or
(3P + S) - (batch tasks)

To really improve the system load, process handling must be extended to an additional level. Consider one main process which creates a menu program for each terminal, redirecting STDIN and STDLIST to the appropriate terminal. Now the equation is:

(d) U + P + S + 2
(2 is the main and its CI)

This equation is only 2 processes over the normal (a) equation, and allows a menu driver for each user. Since the main program controls access from each terminal, security and logon provisions can be added easily. If the attached menu programs are of the multiple type (for concurrent batch tasks), the reduction is even

greater. With no-wait I/O (priviliged), the main task could be a 'global' menu program, controlling all terminals. This eliminates the user menu altogether, giving:

(e) P + S + 2

With this implementation, well over 200 users could be supported on MPE-IV. In general, the process related resource requirements are cut by nearly half. A non-priviliged solution to the above model would be to provide communication paths between the main program and the son processes, then attach the user menu programs. When the function was selected, the request would be sent to the main program and the menu program can terminate. The main program would then attach the appropriate application program. Thus, the main program would alternate between a menu and application processes. The process equation remains the same, but more process switching is required (a suitable tradeoff to remain non-priviliged).

Dependent Processes

Process handling applications involving two or more processes working together for a common goal are the most beneficial forms of process handling. Exact implementations differ on a widespread scale, with only vague suggestions of a concrete definition. Dependent processes are best defined by examination of the basic concepts which can be used. Actual applications usually combine more than one concept, and may utilize only a basic theory rather than a literal definition.

The concept of dependent processes is difficult to define, as some areas are difficult to classify. For example, TDP uses process handling when formatting its output by creating the program SCRIBE to perform the formatting. From one standpoint, the two programs are working together for a common purpose; however, it would be possible to get the same results if SCRIBE was an independent program. TDP is using SCRIBE as a 'sub- routine' to format output, yet this formatting is independent of TDP itself. This arrangement is marginally classified as dependent by the first definition to follow.

Co-routines

A co-routine (usually written coroutine) relationship exists be- tween processes which, when combined, perform a complete task. In strict terms, coroutines are always in pairs, and may ar- bitrarily 'swap' execution from one to the other. A coroutine relationship simply implies a higher level of 'subroutine' than with subprograms. The application of coroutines leads to greater modularity and all of the advantages incurred by modular program- ming. Coroutines may be used to provide greater modularity, or to split a large task (possibly too large or inefficient for a single program) into manageable parts. Standard 'menu' applications described earlier could be considered in this category, but such applications rarely communicate with each other. Coroutines work together for a single transaction, while the applied programs of a menu are usually independent transactions or func- tions. The TDP/SCRIBE relationship can be loosely applied to either definition.

Pipe Processors

The first logical extension of the coroutine relationship is the pipe processor. If the processes of a coroutine are designed so that they accept 'input' from their predecessor, perform some ad- ditional processing, and send 'output' to their successor, the processes have been changed to pipe processors. Programs in this category have an input 'pipe' sending information to be processed, and an output 'pipe' transmitting results. Pipe processors can be hooked together in any number of ways by con- necting the output 'pipe' of one process to the input 'pipe' of another.

One application where this method can be elegantly applied is data base inquiry and report. In this case, there are usually a number of different retrieval methods, sorting methods, and print formats. If each unique retrieval method, sort criteria, and print format was coded as a pipe processor, a flexible reporting system can be created from a minimum number of programs. At run time, the desired retrieval method, sort sequence, and print format are selected and linked together. Each record selected by the retrieval procedure is passed to the sort routine by the pipe. The sort routine passes the records to the SORT procedure until all records are processed. The

sort routine then retrives records from SORT and passes them to the output process by the pipe.

In simple coroutine relationships, these processes would not overlap. Instead of passing data in the pipe, they would trans- fer control to the next process. When the next process completes its handling of the data, control is transferred back to the first process. In other words, the processes must be synchronized together with no overlap. Pipe processors must also be synchronized, but they may overlap. A pipe processor waits until data is available in the pipe, but begins processing once the data is read. When writing to a pipe, the process must wait until the next process reads the data before it loops back for more input. Some overlap is obtained, but the pipe can only hold a single record.

The sorting requirement places a bottleneck in this scheme. All records must be given before the sort process can release one to the report procedure. If the sort is not required, a greater overlap can occur. Without the sort, we have TRANSACTION overlap; with the sort, we have PROCEDURAL overlap. Some applications cannot overlap either case.

Transaction Pipe Processors

When transaction overlap is possible, pipe processors can perform at their most beneficial level. In all pipe processors and coroutines there is some synchronization to be performed. With coroutines, the lack of overlap implies the synchronization, much like a main program calling a subprogram. With pipe processors, there is the question of overlap: After writing results to the output pipe, when does the pipe processor go back for more input? In applications where there is transaction dependency, a transaction may have to finish the entire pipe before the next transaction can begin. In this case, we have no overlap at all. When transactions are independent, the pipe processors can proceed at will, limited only by the capacity of the pipe (in simple pipe processors, the pipe holds only 1 transaction). Overlap is pos- sible, but limited by certain restrictions.

In all cases where a transaction is split into pipe processors (or stages), the processing time for each stage will probably vary from one stage to another. To complicate the matter, the For purposes of present discussion, the average processing time will suffice, and thus the average time for the stages will probably vary.

For example, consider a given transaction divided into four stages. The average processing time for each stage can be denoted by the variables s1 through s4. Regardless of the actual time required for each stage, the time required for each transaction without process handling is:

```
Linear (no process handling):

  s1 + s2 + s3 + s4
```

The use of coroutines will result in a slightly larger transaction time, since the overhead of coroutine switching must be added to the time required for the four individual stages. The same rule applies to pipe processors with transaction dependence.

When transaction independence is possible, the average time for each stage becomes important. If each stage is equal, the time required for each transaction is the time required for a single stage, plus overhead of filling the pipe at first, plus emptying the pipe after the last transaction. The following illustration depicts this case, using 't' for transactions and 'T' for time:

```
Pipe Processor Stages        Time       Linear Processor Stages
  s1   s2   s3   s4           (T)          s1   s2   s3   s4
 ---- ---- ---- ----         ----         ---- ---- ---- ----
  t1                          1            t1
  t2   t1                     2                 t1
  t3   t2   t1                3                      t1
  t4   t3   t2   t1           4                           t1
  t5   t4   t3   t2           5            t2
       t5   t4   t3           6                 t2
            t5   t4           7                      t2
                 t5           8                           t2
```

Of course, this example is an ideal situation. Several factors can interfere with this ideal case: stage times may be unequal, stages may interfere with each other, and stages may have idle time. If the stages are unequal, and the pipe only holds 1 transaction, some stage will obviously be idle for some period of time. This leaves only two main factors to examine: unequal stage times and stage interference.

If stage times are equal, the average transaction processing time is the length of a single stage (once the pipe is filled). If unequal, the transaction processing time is, at best, the length of the longest stage time. If more than 1 stage has an unusually long stage time, and the stages occur in a worst case sequence, the time may be slightly longer. There is, however, a definate overlap of the stages with shorter times.

Stage interference can be present if the stages are divided so that there is contention for some common resource between two or more stages. Examples of such cases include locking a common resource (data base, file, Rin, etc.) and heavily I/O bound stages. CPU bound stages may interfere if the need is excessive (the stage cannot complete before its allocated quantum expires). Most transactions are I/O bound, and CPU bound stages consequently fill in the gaps. If the CPU time for a transaction exceeds the I/O wait time, there is a probability of CPU resource contention; however, the interference caused by other resources is much more exaggerated. Exceptions can be found for most attempts to define rules for preventing resource contention, and to some ex- tent, there is always some interference. The only general guideline is to avoid gross interference cases such as locking.

Proper division of stages to eliminate interference leads to a pipe processor approach which can achieve overlap, which is clearly more efficient than the traditional linear approach. Once this goal is attained, optimization is simplified by con- centrating on the longest stage. However, this approach does not address the problem of data dependent processing time mentioned earlier.

Queued Pipeline Linkage

The next extension to transaction pipe processors involves the means by which the processors are linked, or the pipe itself. If the pipe is a queue instead of a 'mailbox' as assumed earlier, greater overlap is possible, especially when variable stage times are involved. Each stage proceeds at its maximum speed, limited only by the capacity of the queue. The queue effectively smooths out irregularities caused by variable processing times in most cases. The longest stage time still limits throughput, but this approach helps when the longest stage time is variable. Slower stages keep a queue of waiting data, while faster stages remain idle. Interference may be exaggerated by a queued pipeline, especially in I/O bound situations. Queueing will improve over- all throughput (in general) if the straight pipe linkage alone does not saturate the system. It is often necessary to impose some limit of transactions which are allowed in the pipe, espe- cially when pipeline applications affect interactive response time. Most queued pipeline applications can be tuned to a level where the CPU or I/O system is saturated; consequently, other ap- plications on the system will experience poor response time.

All of the previous techniques take advantage of the fact that some idle time is caused during transaction processing. Time which was previously idle is now spent on another task in paral- lel. Although the application under study is experiencing throughput increases, there is a greater load imposed on the sys- tem, and less time is free for other applications. To minimize this impact, the process handling application should take advant- age of resources not currently being utilized. On the other hand, if a given I/O bound application is slow because of the to- tal I/O load of all applications, process handling will only slow everything down. At this level of complexity (and subsequent levels) the tradeoffs in this area determine the benefit or cost of process handling applications.

Switching Pipeline Manager

A switching pipeline manager is a front-end processor for more than one pipe application. In general, if more than one depen- dent process handling application is in use, a switching pipeline manager can be used to route transactions to the proper application. Usually this type of manager controls transaction pipes, and the examination will be limited to this application.

Most systems involve more than one type of transaction, but it is desireable to stay in the same program. An independent process parallel is the menu program, defined in the first section. With a pipeline manager, however, the entire transaction is gathered and 'switched' to the input pipe of the appropriate pipeline. The user may begin the next transaction while the current trans- action is being processed. This is particularly useful in applications which require a printed document as a final result, such as inventory tags, order entry, etc. The user can begin the next transaction while the previous document is being printed. If more than one document is used, the pipeline manager controls the production of the

appropriate one (otherwise a simple pipe processor is sufficient).

The benefit of a switching pipeline manager is limited for online applications, but is extremely useful for batch applications where input (or requests) come from a common source. The next section deals with applying this concept to online systems.

Centralized Pipeline Manager

With online applications, it is obviously wasteful to have a copy of the entire pipeline system for each user. A centralized pipeline manager collects requests from all users of a given system and passes them to a common pipeline system. The collection procedure for the user simply validates the transaction and places it in the input pipe for the centralized manager. In ef- fect, all user collection programs write to a common input pipe (queueing is almost mandatory). If a single application is pipelined with a message file as input, the manager is not really required. A manager process can, if present, control the number of transactions in the pipe and queue the overflow. As mentioned earlier, this limit is frequently necessary.

If more than one application is being pipelined, the centralized manager can be merged with the switching concept described above. In this case, transactions are centralized from the different user terminals and then split according to transaction type. With this setup, it is possible for the manager to provide infor- mation about the number and type of transactions performed for each terminal. Security can be enhanced by limiting transactions by terminal. The manager can even log transactions for recovery purposes without the need for IMAGE or other logging code in the transaction processors. If certain transaction stages require a lot of information to be transferred from one stage to another, the manager can assign extra data segments (from a limited pool) to the transactions. Many functions can be done, but they imply some means of posting the completion of a transaction. This can be easily done by having the last processor in each pipe write a completion message in the manager's input pipe.

Vectored Pipeline Scheduler

In many pipeline applications it is desireable to select certain stages which apply to the transaction. Some pipelines may have common stages, in which case duplication should be avoided. For these cases, the stages can be identified by some method known to the scheduler and the collection program. As a transaction is collected (or received by the scheduler) it is assigned a vector of stages through which it must pass to complete the transaction. In this case, each stage posts completion to the scheduler, which in turn selects the next stage for the transaction. It is possible for each stage to be given selection logic so that sub- sequent stages can be invoked directly, but this causes some redundant code and disables the tracing and monitoring capabilities that can be included in the vector scheduler.

A number of additional functions can be included in the vectored scheduler, particularly when combined with the centralized con- cept previously discussed. The vector can be data dependent, and the vector can be modified by any of the stages. Stage modifica- of the vector can be used for error handling - if a stage detects an error, the vector is changed to pass the transaction to the error handling stage(s). The efficiency of a vector scheduler is much more important than other schedulers due to its intervention between each stage, rather than between transaction tions. For this reason, the vector scheduler usually maintains a high priority queue for stage pipes and a lower priority queue for incoming transactions.

Swapping Vectored Pipeline Scheduler

At the extremely complex end of transaction processing systems we find the swapping vectored pipeline scheduler, an extension of the vectored scheduler defined above. When the number of stages exceeds the number of processes which can be efficiently managed by the system, it is impractical to keep all possible stages loaded at once. The swapping version of the scheduler controls dynamic loading and unloading of stages as required or desired. From a simple requirement standpoint, a preset limit of stages can be determined as the maximum number to be loaded at a given time. As stages are needed, the scheduler fills the pipe (if loaded) or loads the stage. Some replacement method is needed to swap out stages to make room for additional stages when the limit has been reached. The scheduler can be modified to unload stages that are unused for a given time limit.

Swapping schedulers can be used to drive many application sys- tems. This is particularly beneficial when related systems are present. For example, an installation may have systems for in- ventory, order entry, receivables, payables, and ledger. If the systems are broken down into specific stages (or vectors), a single order entry transaction can be transferred through inven- tory (to adjust items ordered), order entry (to print a shipping notice), receivables (for a new bill), and ledger (to post the cash flows between accounts). Furthermore,

the same transaction data can be sent to report stages for queueing daily summary information.

Summary

Many applications of process handling have been examined, some of which are very appealing. A good process handling manager can provide control and communication which are vital for comprehen- sive systems. However, careful consideration should be given to the complexity of the system, limitations of the computer, and error handling facilities. Complex process handling systems pass around a great deal of information, and the possibilities of sys- tem failure, program abort, invalid data, and program detected errors must be taken into account. For example, what does a stage do if an IMAGE error is encountered? What does the scheduler do if a stage aborts?

As mentioned in the introduction, one may swear by or swear at process handling systems. The division is largely up to the process handling manager or scheduler in large systems, but also relates to the resources available in the machine. There are few, if any, instances which require process handling. However, there are instances which may benefit from process handling. In the hands of a careful analyst, it can prove itself to be an in- valuable tool in the design of powerful, efficient transaction processing systems.

Brief Biographical Sketch of Jeffrey R. Kell

Jeff Kell, 25, has been in data processing for eight years, with a background in systems programming on IBM equipment for three years, and five years with Hewlett-Packard equipment. Jeff has been with the University of Tennessee at Chattanooga for six years, and serves as senior systems analyst for Administrative Computing Services. In addition, he maintains configurations, backup, and recovery for the university's five HP-3000 computer systems.

CONFIGURING YOUR SYSTEM

Jeff Kell
University of Tennessee at Chattanooga

What? Me Worry?

System configuration is a highly technical specialty within the realms of systems programming for which there are rarely any concrete guidelines. At times it seems more an art than a science, as any basic configuration will function; to the casual observer there is no readily apparent difference between similar configurations. Only the systems person, eyes aglow like a proud father, seems concerned with such technicalities. With the HP3000, most configuration changes are unnecessary; so why bother?

One unique feature of the HP3000 is the lack of the requirement for an absolute system generation (from scratch) to bring up an operating system. There is no need for the time consuming generation step, the user must only supply the necessary parameters relevant to his installation. If the I/O configuration is omitted from consideration, the remaining items which may be changed rarely disable a system completely. With a few exceptions, the system will run if the I/O configuration is correct. To strengthen the apathy toward configuration changes, the default configuration (or SE supplied configuration) usually works.

A well configured system will outperform a poorly configured system. This benefit is derived from allowing sufficient resources to prevent shortages while trimming the surplus space to a minimum. In this manner, the resident (frozen) memory area is reduced, allowing more memory areas for segment swapping. Slight modification may not produce an obvious result, but overall configuration enhancement will, especially when the system becomes loaded. The results of unoptimized configuration fall into certain categories:

* Table sizes too large:
  - May overflow bank zero; system will not run
  - May waste main memory if table is resident; available memory is reduced
  - Will increase size of table if not resident, and swap time is increased proportionately
* Table sizes too small:
  - If critical, system will not start
  - If serious, increased job load may cause:
    + System failures (critical tables)
    + Job overloads (cannot start new jobs)
    + Failure to :RUN a program (out of resources)
    + Program aborts (unable to obtain resources)
  - If shared, programs are suspended until resources are free

Perhaps the easiest way to present configuration guidelines is to give the usage of each table in a clearer manner than the standard manual definitions. Most tables are user and/or program dependant. In this case the utilization (per unit) will be given for you to adjust to your particular installation. Other tables are relative to other factors, and their definitions will be more concrete.

Code Segment Table (CST)

The primary CST area is used to identify code segments which can be shared by more than one unique PROGRAM, not process (if three users are running EDITOR, there are three processes, but only 1 program). The only entities which can be shared between programs are SL segments (group, PUB, and system). MPE requires approximately 128 entries for itself, depending upon the installed software and I/O devices. I/O device drivers are also given a CST entry (one per driver). The remaining entries are used for user SL segments, one per

segment per SL file. If a segment is present in multiple SL files, the same multiple number of entries will be required to support it. Only segments which are present in the system SL are truly shared across accounts. Insufficient CST entries can block the system from coming up (if very small), cause JOB OVERLOAD messages, and block additional users or programs from execution.

Code Segment Table Extension (CSTX)

The CSTX area is used to identify code segments which can be shared by more than one PROCESS, and are used to map the segments of a program. The CSTX is divided into blocks, mapped by the CST block table, one for each unique program (see Maximum Number of Concurrent Programs). The block for a program contains the same number of entries as there are segments in the program. Subsequent users of the program share the same block. NOTE: if the program uses non-system SL segments, these segments generate entries in the CST table - the CSTX holds only the segments which are physically contained in the program file.

When a program terminates and no other users are executing the same program, the CSTX block is released. Allocated programs are given permanent CSTX blocks which are never released (this is the actual function of :ALLOCATE). The number of CSTX entries required can be calculated as:

```
CSTX = (number of unique program segments)
     + (number of allocated program segments)
     + 15% (to allow for fragmentation)
```

Insufficient CSTX entries block additional programs from execution, usually resulting in a message referring to the CST table, not CSTX. The loader error messages are very vague on this point (see also Maximum Number of Concurrent Programs).

Data Segment Table (DST)

The DST is a rather complicated beast, consuming more entries than anyone seems to expect or can explain. It has been impossible to justify all active DST entries at any given time, but the general tendencies and known quantities of this table can be explored. By definition, there is one entry for each data segment in the system; however, many data segments are generated by the system which are not common knowlege. Approximately 50-60 data segments are used by MPE for system tables, depending on the configuration and MPE version. There is one stack data segment for each process in the system (including MPE system processes), and one entry for each segment acquired through extra data segment capability by user programs. This is where obvious rules terminate.

For each user on the system, without any UDC files (user, account, or system) in use, 4 entries are required just to log on. Of these 4, one is for the command interpreter stack, one for the job directory table (JDT), and one for the job information table (JIT). The other is not immediately obvious, unless it relates to the buffer areas for $STDIN and $STDLIST. If only one UDC is present, two more DST entries are required: one to hold the UDC command directory and one file system buffer segment for the UDC file. Each additional UDC file at any level requires an additional DST entry as a file system buffer segment for file itself. The typical UDC setup with a system UDC and an additional user UDC will require 7 DST entries per user JUST TO LOG ON (64 users = 7*64 = 448 entries).

Along the file system line, one segment is required for each open buffered file. No segments are used by NOBUF files, and some may be saved by MULTI or GMULTI files (which supposedly share buffers). KSAM uses one segment per open file per process, and IMAGE uses one global segment for each open database plus one local segment per DBOPEN per database per process. Since IMAGE works in NOBUF mode, the number of files in the database does not affect the count.

Communications subsystems also use data segments, but the usage will vary between subsystems. DS/3000 uses two segments for each line opened by the operator initially (one stack for DSMON, one control segment), but subsequent closes and opens only affect the control segment (DSMON remains active). Each DSLINE or remote session done by a user requires two segments (possibly a control segment and a file system buffer segment). In an informal approximation, MRJE used 10 entries to support one printer and one punch. Additional printers and punches cause extra processes and extra entries. It is reasonable that the output processes require more segments than just a stack, as they reference the configuration, job, and directory files as well.

In summary, the number of DST entries known is:

```
DST entries = about 50 for MPE tables
            + 1 per system process (stack)
            + 1 per system process extra data segments
            + 4 per user
            + 1 per user with any UDC (including system)
            + 1 per UDC file per user
            + 1 per opened buffered file per process
            + 1 per opened KSAM file per process
            + 1 per opened IMAGE database
            + 1 per DBOPEN per process
            + 2 per DSCONTROL OPENed CS line
            + 2 per DSLINE per user
            + 1 per user process (stack)
           + xx User extra data segment capability
           + xx other communications subsystems
```

Insufficient DST entries can block the system from coming up, cause JOB OVERLOAD messages, block additional sessions or programs, or cause "OUT OF VIRTUAL MEMORY" message on file open attempts (which is another misleading message, although it can be caused by virtual memory shortage).

Process Control Block (PCB)

The PCB table is used to hold information about processes in the system. One entry is required for each MPE system process, one entry per user (session or job), and one for each user process. Spooling processes are included along with MPE system processes, there is one spooler process for each spooled device. There is also a monitor process for each opened communication line, plus any additional control processes required for the communication subsystem in use. The number of PCB entries is:

```
PCB entries = Basic system processes
            + 1 per spooled device
            + 1 per open communication line
            + 1 per MRJE host
            + 1 per MRJE output device (printer/punch)
            + 1 per user (job or session)
            + 1 per user process (:RUN & process handling)
           + xx additional communications processes
```

Input Output Queue (IOQ)

The IOQ is used to hold I/O requests for all non-disc devices. If the number of entries is too low for a given configuration, the system may halt. One entry is used for most terminal read and write requests. One or more entries is used per communication line, and in some instances, by pseudo devices directed to the line. Spooled output devices may use up to 15 entries (10 for input spooling). A good approximation of IOQ limits is:

```
IOQ entries = 1 per non-disc assigned LDEV
            + 10 per spooled input device
            + 15 per spooled output device

256 maximum; may impose limit on spooled devices
```

Disc Request Table (DISKIO)

This table parallels the IOQ for disc devices, but the entries are of different length and format. One entry is required for each process disc request. A minimum of 60 entries is recommended. A process may issue more than 1 request (swapping, file system read ahead, NOWAIT). Insufficient entries will cause a system failure.

Terminal Buffers per port (TBUF)

Older MPE versions asked for this value in terms of a total, but the new configurator multiplies the given number by the number of ports (terminals) currently configured. On non-HPIB systems the buffers hold 30 characters each, HPIB systems hold 62 characters each. In character mode, only one buffer is initially allocated for a pending read; in block mode, all required buffers must be allocated. Note that the specification 'per port' is only for defining a variable total based on the current terminal configuration - the buffers themselves are shared by all terminal devices.

Block mode operations (such as V/3000) are seriously degraded by an insufficient number

of buffers, particularly when concurrent read operations take place. A full screen transfer (1920 bytes) will take 64 buffers for non-HPIB, 31 buffers for HPIB. Note also that V/3000 is not limited to the physical screen, and transfers may be larger. This is particularly true of FORMSPEC, which reads the entire screen at once, not just the unprotected fields. A two-page form (50 lines, 4000 characters) requires 134 buffers on non-HPIB, 65 for HPIB. If block mode is used, the maximum buffers should be configured (the maximum is 255, regardless of the number given per port). Otherwise one or two buffers per port is sufficient.

System Buffers (SBUF)

System buffers are rather vague. Older versions of MPE used these buffers for logon and logoff messages, :TELL buffers, and other things. The current versions use these buffers for I/O error log parameters (intermediary to logging system) and for certain communications line operations. Although the recommended number is 8 in most manuals, some loss of I/O error reporting can occur when tape retries are encountered. Insufficient entries generate warning messages and abort the process responsible for the I/O operation. A better estimate would be 16-18 entries, plus 1 for each MTS line.

Swap Table (SWAPT)

The Swap Table replaces the memory manager table in older ver- sions of MPE, but its function is similar. The number of entries is based on the working set of a process (code and data segments) and is very application dependent. The manual suggests 4 per PCB, although other sources suggest 8, 12, or other values. This table should be watched in order to adequately approximate its size. The suggested value of 4 is considered somewhat marginal, but again, this is application dependent. Insufficient entries will cause a system failure.

Primary and Secondary Message Tables

These tables are used for communication between system processes, the primary table being used for critical messages and the secondary table being used for noncritical messages. Entries in both tables are 5 words long, and the suggested value of 25 is usually generous. A shortage of primary entries is obvious (causing a system failure), but a shortage of secondary entries only impedes the requesting process (which is not obvious or easily measureable).

Special Request Table (SPREQ)

This table is used "to temporarily buffer the parameters for segment expansion and to form the queue for devices waiting for a segment to arrive" according to the System Manager Reference Manual. The memory manager uses this table rather than its own stack to allow size changes. Now that the memory manager overlaps processing with segment I/O, certain things about special segment transfers must be saved (somewhere) for use after the transfer completes. The first case occurs when a data segment expands (a positive ZSIZE or normal expansion during the early life of a program). This occurs infrequently and rarely do requests arrive concurrently unless extensive use of ZSIZE is present in application programs. The second case is not immediately obvious, and the only clue is device related (possibly an I/O driver). The suggested value of 25 is usually generous.

Interrupt Control Stack (ICS)

The ICS is used to process most types of interrupts in the system, and versions of MPE beginning with MPE-IV use this area to process garbage collection by the memory manager. More space is used by these versions than was used previously. The suggested value is 512, and the monitored use of this area is often unpleasantly close to this value (from 475 to 496). One case of power fail recovery problems at UTC was attributed (but not concretely) to the ICS. UTC has subsequently used a higher value, although it has never been monitored over 512. The suggested 512 may be sufficient, but uncomfortable. Allowing some overhead is highly recommended for safety, and a value of 600 is suggested unless proven otherwise.

User Controller Process Queue (UCOP)

This table is used to pass information (requests) to UCOP by other entities. UCOP is documented to manage the initiation and completion of jobs and sessions, and the system device recognition process DEVREC presumably sends initiation requests to UCOP when :HELLO or :JOB is encountered. UCOP also performs other functions as well, such as certain :REPLY operations (a process waiting on a response from PRINTOPREPLY is in a UCOP wait state), deleting processes, changing priorities, and stack size manipulation. The suggested value is 2 per user, with more for process handling. This value has proven to be generous for our installation, and we do an exceptional amount of process handling. Two per user appears to be a maximum, but only 2 words are required for each entry anyway.

Timer Request List (TRL)

This table is used for all timer related functions such as memory logging, the PAUSE intrinsic, timeout and timed reads (terminal and IPC), logon time limits, modem turnaround, and certain timeouts for communication lines. The suggested value is 1 per terminal plus 1 per user, but 1 per user is usually sufficient unless there are a large number of modem ports or communication lines. Standard hardwired configurations can use 1 per user plus 10% for most applications.

Breakpoint Table (BKPT)

This table is used by DEBUG for setting breakpoints in programs. A minimum of 8 to 16 entries will allow for adequate debugging, but more should be configured if DEBUG is used extensively by more than a few users.

Maximum User Logging Processes/Users per Process

These two values determine the number of user logging processes which can be defined and the number of users which may access a given process. Usually one process per application area using logging is sufficient, and the number of users per process is the maximum number of users of any single process (not the total number). This really limits the number of processes, not users, which open the logfile. A large number of users per process increases the space required for user logging control (once initiated).

Number of RIN's

This value determines the number of available Resource Iden- tification Numbers (RIN's) in all (global, local, and file). One per user is usually sufficient to allow for file locking (more if MR applications exist or global RINs are used). IMAGE no longer requires RIN locks to accomplish database locks, thus the 'one per user' is especially true.

Maximum number of concurrent sessions/jobs

These values determine a maximum for the :LIMIT command, and their total determines the size of the Job Process Count table, used when a CPU limitation is imposed on a job. Although not obvious by its query, this value does allocate space.

Logging status (general)

Most installations either allow all logging, or disable all logging. In practice, I/O errors and certain other infrequent status events should be enabled for diagnostic purposes. Logging everything for no purpose clearly burdens the system with unnecessary work. In those cases where logging is important for accounting or security, many logging types are beneficial. Frequent events such as file close and process termination should be questioned for merit, especially in transient environments. File closure, in particular, generates a great deal of data at the completion of every process. System related programs such as EDITOR and the compilers may generate anywhere from 4 to 10 or more records for each completion (a compiler has a record for text, USL, list, master, and new file; COBOL also has work files, symbol table overflow files, and copy libraries).

Virtual Memory

Virtual memory allocation is best spread across volumes, although one could argue merits of other schemes. The technique is not the purpose of this discussion, so it will be left to rest. If virtual allocation is spread, however, it should be reasonably close to the amount actually used; otherwise the used area may reside on a single volume and probably will be on the system disc. Virtual memory is only used to store data segments, not code segments. Code segments are swapped directly from program and SL files.

The amount of virtual used will be, at a minimum, the size of all data segments. Each segment requires a number of sectors equal to the segment size in words divided by 128, adding one unless the remainder was zero. The size of a data segment is fixed by GETDSEG, and the size of a stack is the MAXDATA value plus the system defined area at the beginning of the stack (PCBX). Some shops use a large default stack size, and MAXDATA is allowed to default (See section on Standard Stack Size). Insufficient virtual memory causes any process requiring an additional segment to be aborted (:RUN, process handling CREATE, and file opens).

Maximum Number of Concurrent Running Programs

This value determines the size of the CST Block table (CSTBLK) used in conjunction with the CST Extension. This is NOT related to the number of processes or users, but refers to the number of unique programs which may execute concurrently (additional users of the same program do not count). Insufficient numbers cause :RUN to fail, usually with 'Unable to obtain CST entries' error, which does not immediately indicate the problem. This value does not necessarily have to be large enough to prevent the message - it is frequently necessary to limit this value for performance purposes. Additional users of a program already loaded do not impact the system as much as new programs do. Note that :ALLOCATED programs take up an entry, as :ALLOCATE simply assigns a permanent entry to the designated program. The system will add 8 to the value you define to account for system programs used by MPE.

Maximum Code Segment Size/Maximum Code Segments per Process

These values place restrictions on programs. Once upon a time, HP said 4K segments were adequate for all of their products, but this value has increased to 8K. Arguments can be made for and against larger segments, but it should be noted that there are contributed library programs which have segments larger than 8K. In a controlled environment, it may be adviseable to set this value higher and limit the size by individual control. In an uncontrolled environment, this is not practical, and the limit should be set to 8K. The number of segments per process is suggested to be 63, smaller values may cause problems with large products (compilers, etc.). This only limits the eventual size of a program, and has little if any impact on performance.

Maximum Stack Size

This value limits the size of a stack, not to exceed 31232. Smaller limits may be argued to help performance by decreasing swap time, but many products (V/3000, SORT) are either adversely affected or cannot run altogether with smaller stacks. As with code segments, it is adviseable to set this value to its maximum and control sizes by some manual guidelines.

Maximum Extra Data Segment Size

If this value is large, and little control is exerted over data segment management, this may lead to performance degradation. This value affects IMAGE directly by limiting the buffers available in the global control block. If this size is too small, IMMAGE may not be able to allocate enough buffers to be helpful (the segment allocates a certain amount for control information and lock information - this amount must be subtracted from the maximum size to determine the space available for buffers). If this segment limit is too low, it will be impossible to open buffered files with large block sizes. HP recommends 8K for some reason, but IMAGE and extra data segment capability are the only areas which can misuse this limit. As with the previous limits, it may be adviseable to configure the maximum limit and control the actual use by other means.

Standard Stack Size

This parameter has perhaps the greatest misuse potential of all in terms of performance. It is used to supply a system default for the initial stack size (Q-Z) if the user does not override it. Some installations will set this to a large value (4K or more) to avoid run time STACK OVERFLOW and SORT: INSUFFICIENT STACK SPACE errors. Most programs, except for V/3000, SORT, and MERGE applications, will run with a default stack between 1600 and 2000. This situation deserves clarification.

When a program is loaded (with default STACK and MAXDATA), the data area defined by the program globally is already allocated. This data goes into the DB area, and the final position, plus one for PARM and a variable area at run time for INFO, determines the location of Q. Below DB, 128 words are reserved by default unless a DLSIZE is given. Below this, the system has a reserved area. This system area is not included in any specifications, but is subtracted from the 32K possible words to derive the 31232 maximum stack size possible. The area above Q is used for procedure storage (or for any called subprogram data in COBOL). Typical FORTRAN, SPL, or PASCAL procedures will not use much local storage, but COBOL is more likely to use a bit more. Most standard MPE intrinsics do not need a great deal of storage either. FOPEN is a typical case taken as a maximum, requiring around 1200-1400 words (more for labeled tape and laser printer, and remote file access). The point to be made is that a default stack size of about 2000 words (suggested in manual) is more than adequate for most applications.

If MAXDATA is not specified, it defaults to:

```
MAXDATA = DLSIZE (128 if not specified)
        + DB-Q area (from program)
        + STACK (uses Standard Stack Size if not given)
```

The Z register is planted at the end of the STACK area, and the final size MAXDATA + (system reserved area) determines the data segment size of the stack. This same area is reserved in virtual memory for the stack. Swapping occurs from the beginning of the segment to the location of Z. If MAXDATA is specified, Z is still placed at the same location, but virtual memory is reserved for MAXDATA. In other words, the segment swapping is directly related to STACK, not MAXDATA itself. During execution, if the stack reaches Z, it will expand automatically if more space is available, but will cause overflow if MAXDATA is exceeded.

SORT requires additional stack space for a work area. V/3000 uses a good deal of space in the stack as well, but also uses the DL area. Both products may fail with a default stack and no override MAXDATA. Changing the STACK parameter will help SORT, and usually help V/3000, but MAXDATA does the job also. If the installation standard stack size is changed to help these areas, everyone suffers. If a program does not need this extra space, it is not only wasted in memory, but the unused space is also swapped.

MAXDATA is better than STACK, and should be used instead. although not recommended for everything, it would be better to have a system UDC to PREP everything with a large MAXDATA than to change the default stack. MAXDATA wastes only virtual memory, but STACK wastes virtual memory, real memory, and swap time. We firmly believe that a standard stack size of 2000 (or slightly less) is adequate. In other terms, it is better to override small defaults for large sizes than to override large defaults for small (programmers will not go to the trouble to change something that doesn't abort their program).

Maximum Number of Open Spoolfiles

This parameter is asked before any tables, but listed here to keep the obvious tables together. Usually no connection is made between spooling parameters and memory usage, but this parameter does indeed allocate permanently resident storage. Most installations will use a high value for spool file kilosectors so that spooling is limited only by available disc space. This practice is acceptable, but the value for open spoolfiles should be minimized. Spool files in the READY or ACTIVE state are unaffected, only those OPENED at a given time are affected by this parameter.

When a spool file is opened, MPE sets up a dummy device in the location of an unused logical device number. This parameter tells MPE how many dummy device numbers to reserve for spooling. Each entry requires 8 words, and the maximum is determined by the number of real logical devices on the system:

$$\text{Maximum value} = \frac{16244 - (\#\ \text{of LDEV's} * 4)}{30}$$

Configuration Table Parameters Summary

Table Name	Entry Size	Resident	Min	Max	Insufficient
DST	4 words	Yes	192		Oflow; load
CST	4 words	Yes	93	192	Can't boot
CSTX	4 words	Yes	32		Oflow; load
PCB	16 words	Yes	?	256	Overflow
DSKIO	16 words	Yes	?	255	SF601
IOQ	11 words	Yes	20	256	Impeded
TBUF	16/32	Yes	?	256	Impeded
SBUF	129 words	Yes	?	?	Impeded
SWAPT	5 words	Yes	?	2048	SF602
Primary Msg	5 words	Yes	?	255	SF620
Secondary Msg	5 words	Yes	?	255	Impeded
SPREQ	5 words	Yes	?	255	SF600
ICS	given	Yes	?	?	SF/Halt
UCOP	2 words	No	?	?	Impeded
TRL	4 words	Yes	?	?	Fail
Breakpoints	5-12	No	?	?	Can't set
# User log procs					
# Users per/log					
Rins	2 words	No	5	?	Can't use
Global Rins	12 words	No	?	?	Can't use
Max Sess/Max Job	1 word	Yes	?	?	Overflow
Max Programs	?	?	?	?	Can't run
Max open spoofles	8 words	Yes	?	?	Can't open

Brief Biographical Sketch of Jeffrey R. Kell

Jeff Kell, 25, has been in data processing for eight years, with a background in systems programming on IBM equipment for three years, and five years with Hewlett-Packard equipment. Jeff has been with the University of Tennessee at Chattanooga for six years, and serves as senior systems analyst for Administrative Computing Services. In addition, he maintains configurations, backup, and recovery for the university's five HP-3000 computer systems.

'Data Communications Shortens the Distance'

James F. Dowling
Bose Corporation

ABSTRACT

Bose has recently implemented three new data communications services; an HP2333 Cluster Controller, a MICOM Micro-600 Port Selector, and a data communications feature for our ROLM CBX. This paper will describe each of these systems, their capabilities and shortcomings and operational considerations. Details of cabling, configuration, modems, user interface and system documentation will also be presented. Additional details including; how to survive a lightning strike, data communica- tions troubleshooting and tools of the trade will be presented.

Introduction

During the past six years Bose Corporation has outgrown a Series II, a Series III, a Series III plus a Series 33 and now we are upgrading from two Series 44's to a Series 48 and a Series 68. In this period, we have expanded service from six users to seventy users, ten of which are lo- cated at a manufacturing plant that is ten miles away. User require- ments have forced us to include dial in access, dial-out access and Remote Job Entry to a service bureau. Along the way we have reviewed our local terminal cabling several times and have gone through as many schemes for combatting lightning and other electric disturbances. Now that we have experienced all of the possible problems and finally deter- mined once and for all what the user requirements will be for the next three to five years; we feel that we are set to construct the ideal net- work (dream on ...) Actually, what we have done is, for the first time, designed a system that integrates data communications components with our voice communications systems provide a flexible network. I feel obliged to state that at the time of this writing, our new data center is still under construction and we are operating with a mixture of old and new systems. Until we have completed the conversion we will be un- able to draw a final conclusion. We have however demonstrated the func- tion and value of all of the techniques and components that will be presented below.

The Nodes

- Terminals
  - (40) HP2640B
  - (20) HP2624B
  - (12) DIRECT 825
  - (15) HP2645(9)A
  - (3) HP2626W
  - (2) HP2623A
- Personal Computers
  - (3) HP120
  - (10) HP150
- Computers
  - (2) HP3000 Series 48 (Now)
  - (1) HP3000 Series 48 (Soon)
  - (1) HP3000 Series 68 (Soon)
  - (1) Prime 400
- Printers
  - (1) Printronix P300
  - (4) HP2631B
  - (2) HP2631A
  - (1) HP2601A
  - (2) HP2619A
  - (1) HP2617A
- Miscellaneous
  - (2) Command Port on ROLM CBX
  - (1) Command Port on MICOM 600
  - (1) Stats Port on MICOM 600
  - (3) VADIC 300/1200 Baud Modem
  - (2) VADIC Autodial Modem
  - (4) ROLM CBX Data Port

The Links

- CPU to CPU between the HP3000's
- Local terminals to both CPU's
- Remote terminals to one of the CPU's
- Local spooled printers
- Remote spooled printers
- Out and In Modem access
- PC to PC
- PC to both computers

The Network Components and Connections

CPU to CPU -

This is accomplished with the HP DSN/DS software, an INP in each machine and a cable between the two. Even when you add the I/O configuration step, this is a simple connection to establish. DS gives us the ability to transfer files between our two HP3000's and to execute Sessions on one machine when a terminal is connected to the other. We have experienced tremendous throughput degradation over the DS line when more than three users are working. We also find that CPU loading on the two ma- chines can effect performance. A general statement would be that for quick (tapeless) file transfers it is indespensible but for remote sys- tem access we need a better solution. There are a few things to con- sider when configuring your DS Lines. Because each site is different I suggest that you consult with your SE Data Comm Specialist but, here are some suggestions:

. Use meaningful device class names eg. Our line from System 1 to System 2 has the device class names DSLINE and SYSTEM2 DSLINE is meaningful to your SE and CE while your users can access it with the comand :DSLINE SYSTEM2.

. Never tell the users the Logical Device #. They will code it into something critical that will blow up the next time you do an I/O configuration change.

. Make sure that your block length is compatible with all HP Soft- ware. HPSLATE, DEL, V/3000 and other products require larger than "suggested" block size.

. Close DS Lines before a SHUTDOWN; your system could HANG or the code that is downloaded to the INP could be corrupted. We have even seen the remote machine hang when a system is shut without first closing the line.

. Watch for I/O errors and disconnects in your system logfiles. We found that a disconnect is logged every 30 seconds if the CPU loading gets below 10%. We also detected a fault on an otherwise normal line and traced it to a defective INP by noting a sudden occurrance of Recoverable I/O errors.

Local Terminal Connections

As stated above, we found that the DS line between our two machines was inadequate. We used stock A.B switches for some of the more frequent "System Switchers" but the waste of leaving a much sought after port unused 50% of the time was unacceptable. To solve the problem we have installed a MICOM Micro 600 Port Selector which allows the terminal user to select and queue for ports on all configured machines. More will be said about the MICOM later.

Connecting HP terminals to the HP3000 computers should be a simple task but we have found a myriad of ways to mess it up. The easiest way is to decide to save money and buy the cables from a supply house or even better, make them ourselves. Well lo and behold the HP262X Series that uses a new connector or the cable from a third party that has TX and RD transposed. The wait for parts or cables is an avoidable embarassment. Following is a description of our scheme for connecting "Hardwired" ter- minals to the computers:

. At the terminal end use a Lightning Protected Modem Cable pur- chased from HP. The lightning protect cables have greatly reduced our demand for replacement "ASYNC DC" boards for our 264X ter- minals. The MODEM version is used so that a user could easily install a switch box to an Acoustic Coupler or Modem. Note that we are now able to carry all required signals to a modem if desired. If we used the less expensive "Direct Connect" cable we could not.

. We use standard (BELL Code) "Station" wiring from the user loca- tion to the computer room. We have run two four-wire lines from the offices to "Closets" in the office areas. These are cut down to standard telephone blocks in the closet. This gives us cabling for voice and data to spare at each location. In the office, we connect the terminal to the line with a homemade

RS232C to tele- phone block cable. Eight or four wire modular connectors can also be used and in some locations, wall mounted RS232C connectors are used.

. To get from the "Closet" to the Computer Room "Frame" we run a 50 or 100 pair "Feed" cable cut down to 50 pair blocks on each end. Each of these blocks is equipped with two 25 pin connectors so that all lines can be accessed by a cable or by cross-wiring. The connection from terminal to Computer Room is completed by "cross-wiring" from the "Station" block to the "Feed" block. The use of Station to Feed cross connecting has several advantages.
  - Telephone technicians understand it.
  - Hardware is inexpensive, reliable and available.
  - When a terminal moves, we don't have a useless cable running to the Computer Room.
  - New installations generally require no cable pulling and can be done in minutes.

. The Computer Room "Frame" consists of two fields of standard tele- phone cable blocks. One field is feed cables from the closets bringing the terminal lines in. The other field is computer ports. A cross-wire connection is made here and we are almost to the CPU.

. Having two machines with a total of 64 ports and looking forward to 128 ports in a year or so, we decided that 128 cables across the floor was not a good prospect. Over the years we went from "exact length" to "leave two feet slack" to "buy thirty footers" as our purchasing guide. Our solution was to have some cables made for us. The cable we call HYDRA has six RS232C connectors on one end, a twenty-five pair cable in the middle and a standard TELCO fifty pin connector at the other end. The cable is only four feet long so to get to the Frame we use standard extension cables of the appropriate length which plug into the cable blocks. With one cable for six terminals everything is much cleaner and more reliable.

. What remains is the I/O Configuration. Here are some hints that we find useful.
  - Assign meaningful device class names such as HWTERM for hard- wired, PHTERM for modems, HP2631B & REMOTE1 for a spooled remote printer. These names can help a lot when you forget a logical device number or when a port fails and you must reconfigure.
  - You must use fixed speed if you want to be able to log on at 4800 or 9600 Baud.

. Your system console port (LDEV20) is a strange one. We discovered that an MPE gremlin can alter your I/O Configuration for LDEV 20. Here is the gotcha: You have your console configured for fixed speed at 9600 Baud; You slow to 1200 Baud for Remote Diagnostics; System Crash; Warmstart (Console is at 1200 Baud); Speed up to 4800 Baud; Shut Down; Coolstart; System will not start. If you now change the terminal speed to 1200 Baud the Coolstart will work and if you look at your I/O Configuration you will find that LDEV 20 has been changed to 1200 Baud. This is an MPE Bug; watch for it.

. Another consideration is the remote dignostics port (LDEV 21). This port should be configured for and connected to one of the HP approved modems. When not in use by HP for diagnostics it can be used for dial access to your machine. See below for more information on modems and security. Here are a couple of things to consider:
  - The diagnostic computer (CMP) has it's own SPEED command and CMP speed is set equal to that of the CONSOLE (LDEV 20) at the last System Start (See above for complication here) therefore, you should change your Console speed and then CMP speed to that of the modem before you enable remote diagnostic access.
  - The CMP presumes that it is connected to an HP terminal. Consequently it uses ENQ/ACK protocol for flow control. If your CONSOLE is not configured for ENQ/ACK, you will ex- perience a long pause every so often while MPE waits for the terminal to respond to a flow control request.
  - We have also found that if Remote Diagnostics is enabled and the modem is disconnected two

phenomenon occur. First; a running system can HANG and Second; a System HALT may occur after responding to the "Date & Time OK?" prompt at system startup.

Remote Terminals

We currently use an eight channel multiplexor and a twelve channel Cluster Controller (HP2333A) to connect 11 terminals and three printers to one of our systems. We started out using only the multiplexor for both terminals and printers. This was an interrim step with DSN/MTS selected as the permanent solution. While we were planning the MTS net- work we learned about the Cluster Controller. The cost of the MTS in- terfaces and special cables combined with the potential user confusion (Enter v.s. Return) were already major concerns. The HP2333A looked like a better solution with few disadvantages except that we would prob- ably receive serial number six and all of its problems. Since the multiplexor would continue to work ok for awhile, we discussed the HP2333A features and functions with the HP factory people, ordered one and hoped for the best. Our experience with the system has been very good and user sattisfaction is high. Running 10 terminals at 4800 Baud simultaneously over a single 9600 Baud leased line we find no percep- tible delays. Both Block and Character mode transmissions work equally well.

In comparing Multiplexor, Multipoint or Cluster options the following considerations arose:

. Cost: The multipoint system was the most expensive due to the terminal interfaces and special cabling. The Multiplexor system was least expensive and the Cluster Controller fell in between.

. User Satisfaction: The Multiplexor offered the least desirable solution due to noticable delays on both transmission and receipt of data. The Multipoint system was responsive but there were several software products that would not run over MTS/Multipoint and the "ENTER v.s. RETURN" conversion presented a potential for confusion. The Cluster Controller gave our remote users a system that was equivalent to that of our local users.

. System Administration: Obviously the I/O Configuration, MTS con- figuration and console commands make the Multipoint and Cluster Controllers more difficult to use than multiplexors.

Data Integrity: Multipoint offers the highest degree of data in- tegrity. Effectively data is error checked and retransmitted if necessary between the CPU and the Terminal. The Cluster Controller provides the same except that only the CPU to Cluster Controller link is checked. Data errors occuring on the line between the Terminal and the Controller will go undetected. The Multiplexor link will only control errors that occur between the two Modems leaving both end links unprotected.

Local Spooled Printers

There are three types of "Spoolable" printers on our systems; HP-IB con- nected, Serial (ADCC) connected and Parallel Differential (26069A I/F Board) connected. We use our HP2608A in a standard configuration for local (less than 12 meters) printing. We have an HP2631B spooled through an ADCC in a standard configuration (See Communicator #30 for some important configuration details). One of our HP2619A clones (actu- ally they are converted Chain Trains that had Data Products interfaces) is shared by both systems. To do so we purchased an automatic switch box that will toggle back and forth between the machines looking for a printer request. It will alternately service System 1 and System 2. Using DSN/DS and remote file access is much less efficient and is more cumbersome. We have also been succesful sharing ADCC connected printers using a "modem sharing" device. We attempted to use an RS232C Serial printer spooler like one would use with a Personal Computer but found that none would support the rather unique flow and status controls that HP incorporated into the HP2631B Remote Spooled Printer Option. As with most other aspects of the HP3000 we've developed a few hints:

. The HP2631B can, at best, print 180CPS. If configured at 2400 Baud you will experience some CPU and I/O overhead due to repeated attempts to add to the printer's buffer. Try using 1200 Baud if you can to minimize the wasted I/O and CPU and suffer little in printer throughput.

. The HP2631B can print 225 characters on standard paper in compressed mode. Unfortunately the SPOOLER reset the option to 10 CPI as it closes each SPOOFLE. Your local SE can give you a patch to defeat this if you desire.

Remote Spooled Printers

The HP2631B makes a good remote spooled printer. It responds to Status requests with Offline, Disconnected and Paper Out indications but can not sense a paper jam. We have ours connected to one channel on an eight channel multiplexor that has an integral 9600 Baud modem which in turn connects via a leased line to a remote plant. Since we were carry- ing two HP2631B printers and six terminals on the same multiplexor we chose a MUX that had a printer priority control feature incorporated. This feature prevents the printers from consuming an excessive percentage of the bandwidth during heavy print loading. I might note that the DSN/MTS system does not have such a feature and significant degrada- tion can occur when the printer is busy.

It is possible to use an HP2608A as a remote, spooled printer even though it is an HP-IB device. HP sells three models of a device called an HP-IB extender. One option allows remote connection via coaxial cable, another uses fiber optics and third uses telephone circuits either switched or non-switched. A local user has an HP2680A Laser printer operating on the coaxial cable version. We have ordered the same for our HP2608A. The only drawback to this system is that the local station can not sense a power failure at the remote end and will subsequently send print to never never land.

Modems

Bose uses a ROLM CBX for internal telephones and access to the Bell Net- work. All of our dial access modems are connected to the internal ex- tension network rather than to Bell lines. This gives us many advantages:

- Security: To access the system you must request to be connected to the appropriate modem. This allows us to change extensions at will and eliminates the Random Try for a data line by a hacker. Changing Bell phone numbers is much more complicated and less secure.
- Flexibility: We can add modems without the Telephone Company delays.
- Features: You can outdial from an answer only modem by calling out on one phone to the destination then transfering the call to the local modem.
- Cost: By using the same lines as the voice systems do we can take advantage of the "Least Cost Routing" feature of the telephone switch. We also avoid all interconnect and dedicated line costs.
- Reliability: With a dedicated data line you count on it working all the time. By using our Private Switch we can get in or out on any of the available trunks.

Following are a few things to keep in mind when configuring modems:

- If you use a PBX that has a "Call Waiting" tone you should have this defeated or your Modems will lose data each time the tone is received.
- Use 1200 Baud and Speed Sensing even for 300 Baud modems to allow backup for 1200 Baud modems without a shutdown.
- If you use hardwired (Subtype 0) rather than switched (eg. Subtype 1) for modems you must strap DTR high for the modem to make a connection. The result of this is that if the caller does not log off (:BYE) his session will be left active on the Port and the next caller to that modem will be able to pick up where he left off. For security this is not desirable but for special cases such as data entry users or printers it may be useful.

The ROLM CBX Data Feature

ROLM offers a Data Communications Option for its Computerized Branch Exchange (CBX). This system allows a user to connect Terminals, Ports, Modems and Printers to the CBX in much the same manner as telephones are. Each device is connected to a Data Terminal Interface (DTI) which is located at the device. Standard telephone cabling is used to carry the data to a Data Line Interface (DLI) which is installed in the tele- phone switch. The CBX then provides a means of assigning ports to groups such as SYS1 and SYS2. The computer is then "called" by the ter- minal through a setup dialog where the user enters a port group name. Available port selection or queuing for one will then be managed by the CBX. Contrary to the advertising this does not use "Current Telephone Wires". Each DTI must have its own 3 pair cable.

We investigated the use of this feature at Bose as a part of an overall networking and computer hardware plan. The following features of the ROLM package were incorporated into our plan:

Use an outboard device to manage access to the computers. This offers an alternative to using DSN/DS for Remote System Access.

. The Port Access Controller should provide queueing for ports. This allows us to configure the computers with only as many ports as we will allow simultaneous sessions on each.

After a thorough review of the ROLM Data Feature we decided that it (or something similar to it) should become the hub of our local data com- munications network. Although this system provided all of the critical features that we desired, we were disappointed to find several major inconveniences:

Each Port and Terminal is connected to the Switch though a small box called a DTI located at the device. To make a connection to the switch the user presses a switch on the DTI. At the terminal end this was acceptable since the box could sit under a telephone. On the other hand we worked for hours trying to figure out how to house up to 150 of these little boxes in our computer room. The availability of a Rack Mounted Card Cage version would have been ac- ceptable; the shelving to store the DTI's was not.

When setting up a connection at your terminal the CBX communicates at a data speed no greater than 100 CPS. This is painful to say the least.

When assigning ports to users the switch issues them in "Round-Robin" order. The result of this is that statistically all ports will ap- pear to be used equally. This presents a problem if you desire to know how many ports you need to handle the traffic. We would have preferred a "Hunt" method. This would statistically force the first ports in the group to be used the most and successive ports to be used less and less thereby giving a clear picture of traffic.

An incredible omission in the ROLM switch is that it does no logging of connects and disconnects. This leaves the System Manager without any data to analyze system loading or effectivity.

Our study of the ROLM Data Communications feature whetted our appetite for an intelligent Port Access Controller leading us to investigate Develcon, Gandalf and MICOM as alternatives. The basis for our selec- tion of a MICOM/ROLM system is material for another paper but it suf- fices to state that we have incorporated this combination of two systems into our network. The MICOM handles all local terminals and incoming modems while the ROLM will be used to handle outgoing modem lines and sòme special cases. We feel that as the ROLM system matures it will become a more significant component in our system.

The MICOM Micro 600 Port Selector

For simplicity the M600 placement has been omitted from all of the above details. Actually most of the devices that are connected to the two computers do so through the M600. If you refer back to the section above that describes our computer room wiring Frame you will note that it was described as having two fields of wiring blocks. Actually there are three (I apologize for the lie but believe me it was for your good and my sanity); one is for the incoming terminal lines; the second is for the computer ports and the third is for the MICOM M600. The port and line connections from the M600 connect to the third (physically cen- tral) field by using, again, standard telephone 25 pair cable extensions with connectors (now you know where the idea came from). Terminals that go to the M600 are cross-wired to the center field and those that will be direct-connected are cross-wired directly to the computer port field.

The function of the M600 is essentially the same as that of the ROLM. The terminal user connects to the M600 by typing a Carriage Return (or space bar at 4800 Baud). The M600 prompts for a port class and after accepting it from the user's keyboard it searches for a port of the ap- propriate speed. If none are available, the user is shown the current queue length and is given the option of being added to the queue or dis- connecting to make another class. The M600 comes with the interfaces mounted four to a circuit card rather than in little boxes; it sets up connections at full terminal speed and connects and disconnects are logged through an RS232 port. Unfortunately the M600 also uses a Round-Robin port selection algorithm. Without making any further comparisons between the MICOM and ROLM system, following are some ideas and tech- niques that we have found useful:

. Configure the computer ports as full duplex modems using the switched line option (Sub type 1). This will solve two problems for you. It will cause the session on the computer to be aborted if the M600 fails, if the M600 operator forces a disconnect or if the terminal user disconnects. It will also cause the port and terminal to be disconnected from the M600 if the computer fails.

. The M600 can be configured so that idle ports or terminals will be automatically disconnected after a preset time interval from 1 to 255 minutes. This will return unused ports for reassignment but may cause problems

for terminal users unless the time period is made consistent with user needs. We use one hour on our develop- ment computer and still have some surprises such as a disconnect while awaiting a tape mount or during an FCOPY of 250,000 records across a DSN/DS Line or while printing a long report to a remote printer.

. HPWORD requires that workstations and printers be connected to specific logical device numbers. This can be achieved through the M600 by assign- ing these ports unique class names or by using the M600 console to force a connection between the device and its as- sociated port. The former is prefer- red for the work stations be- cause it allows them to access other resources but the latter is necessary for the printer because they cannot initiate the connection.

. Our M600 Command (Console) Port is connected as a port on the M600 allow- ing any terminal (who knows the class name) to access it. We use 1200 Baud for the Command Port so that modem access is available.

. The M600 Statistics Port (logged con- nects & disconnects) is also connected as a Port so that it can be monitored from any terminal. The Statistics Port is connected in parallel (See Reference #16) to a printer for hard copy. We plan to log this to some storage device once that system is defined. The Stats Port is also con- figured at 1200 Baud for modem access.

. It is possible to connect Computer Ports as M600 Lines. This al- lows the computer (with some user software) to connect itself to other devices either connected as Lines or Ports. This can be useful with Auto-dial modems to allow the CPU to connect to a remote printer.

. It is possible to connect terminals or printers as M600 Ports. This will al- low users to select the device and "at- tach" it to his terminal. A printer or terminal in the computer room can be used as a message drop in this manner.

. Connecting from terminal to terminal is called Matrix Switching on the M600. This feature unfortunately does not use class names rather you request a specific Line number (and hope that it hasn't been reassigned) to connect to another terminal. This is quite valuable in a network with many Personal Computers as it allows PC to PC connection without modems or extra cables.

We have found the M600 to be a flexible data communications switch in both manual and interactive modes. It has helped to simplify the onerous chore of cabling and connecting terminals to our computers as user needs change while trying to keep the documentation up to date. The M600 hardware and software are continuing to ad- vance bringing in- creased value to our system.

Conclusion

The distance between the Computer Operations Group and the computer sys- tem users may be measured physically in hundreds of feet or in miles but the emotional distance is measured in degree of satisfaction. The data communica- tions system that connects the two will determine if not only limit how close they may get and stay. The needs of the system user are specified in terms of how fast the terminal is, terminal relocation de- lays, new installations, up time, system availibility and response time. The system designer must add cost, com- plexity, backup, data integrity and main- tainability. The right equipment for most networks is readily available, it may be hard to find or even recognize but it is probably out there. We at Bose spend a significant amount of time looking at how others solve their problems, we learn from their experience then we determine the applicability of their solutions to our situation. In some cases, such as the HP2333A Cluster Controller, we must rely heavily on a vendor's reputation and per- sonal evaluations when selecting a solu- tion. In either case it is important to keep the deci- sion process open to change long enough to assimilate data from as many sources as pos- sible; Use as many proven components as possible in critical situa- tions; Provide fal- lback systems for all components or systems and above all try to make your users think of the cables, modems, multiplexors etc. as part of a system not an obstacle that is between them and get- ting their job done.

I wish you success in your Data Communica- tion Networking and hope that you have found some useful information in this descrip- tion of our (cur- rent) network.

References

Texts and Handbooks

1. Data Communications, A user's Hand- book, Racal Vadic

2. Standards and Protocols for Com- munications Networks, Conrad, Car- negie Press, 1982

3. McGraw-Hill's Compiliation of Data Communications Standards, Folts et. al. McGraw-Hill, 1978

4. Computer Networks and Their Protocols, Davies, et. al., Wiley & Sons, 1979

5. Distributed processing and Data Communications, McGlynn, Wiley & Sons, 1978

6. Data Communications Facilities, networks and Systems Design, Doll, Wiley & Sons, 1978

7. Telecommunications and the Computer, Martin, Prentice Hall, 1976

Hewlett Packard Manuals and Guide Books

8. HP2645A Display Station Reference Manual, Hewlett Packard, 1978

9. HP3000 Computer Systems Handbook, Hewlett Packard, 1981, Part #30000-90105

10. Guidebook to Data Communications Hewlett Packard, 1977, Part #5955-1715

11. Introducing the 2631B Remote Spooled Printer, Ishida, Communicator #25, 1980, Part #5951-6113-25

12. HP3000 Asynchronous Serial I/O Printer Support, Couch, Communicator #30, 1982, Part #5951-6113-30

13. Hewlett Packard Electronic Instruments and Systems Catalog, Hewlett Packard, 1983, (Reference for HP-IB Extender)

Technical Papers and Articles From HPIUG Publications

14. Everything You Wanted to Know About Interfacing to the HP3000 - The Inside Story, Scroggs, Proceedings of the 1983 International Con- ference (Montreal), 1983

15. Lightning Transients and the RS-232 Interface, Habron, Journal HP General Systems Users Group, Vol. III No. 3, 1980

16. Poor Man's Multidrop, Beckett, Journal of the HP General Systems, Users Group, Vol. 2 No. 1, 1978

17. Asynchronous Communications Protocols, Villa, Journal of the HP General Systems Users Group, Vol. 1 No. 6, 1978

18. Lightning and Your Communications Lines, Hartage, Journal of the HP General Systems Users Group, Vol. 1, No. 6, 1978

19. Life at the End of a Phone Line, Crow, Proceedings of the HP General Systems Users Group 1981 International Meeting (Orlando), 1981

20. Pseudo Devices, Primmer, Proceedings of the HP General Systems Users Group 1981 International Meeting (Orlando), 1981

21. Local Area Networking Simplified with a Data PABX, Skaug, Proceed- ings of HPIUG 1982 International Conference (Copenhagen), 1982

22. Performance Characteristics of HP/DSN (DS-3000), Brawn, Proceedings of the 1983 HP3000 International Conference (Montreal), 1983

23. Synchronous Communications on the HP3000, Demos, Proceedings of the 1983 HP3000 International Conference (Montreal), 1983.

Sources of Equipment and Supplies

24. Business Computer Concepts, 108 Kirkwood Drive, Mars, PA 16046, (412) 962-5993 HP2619 Printer Switch, HP-IB Bus Switch, Custom Data Comm Equipment

25. INMAC, 130 S. Wolfe Road, Synnyvale, CA 94086, (408) 737-777 Cables, Switch Boxes and Supplies

26. Gandalf Data Inc., 12301 Old Columbia Pike, Silver Springs, MD 20904, (301) 622-5400 (Eastern Sales Region) Modems, Multiplexors, Port Selectors and Line Drivers

27. Develcon Electronics Inc., 4037 Swamp Road, Doylestown, PA 18901, (215) 384-1900 Port Selector

28. MICOM, 20151 Nordhoff Street, Chatsworth CA 91311, (213) 998-8844 Modems, Multiplexors, Line Drivers and Port Selectors

29. ROLM (New England), 125 Hartwell Avenue, Lexington, MA 02173, (617) 861-0730 CBX and its Data Communications Feature

30. Black Box Corportaion, Box 12800, Pittsburgh, PA 15241, (412) 746-2910 Cables, Switch Boxes, TELCO Wiring Supplies, Specialty Items to solve dozens of Data Comm Problems

31. Solana Electronics, 249 South Highway 101, Solana Beach, CA 92075, (619) 481-6384 Inexpensive eight channel multiplexor/ line driver unit for use wthin a facility (No FCC Interface Clearance)

32. Hewlett Packard Co., 1280 Embarcadero Road, Palo Alto, CA 94303 HP3000 Products, HP-IB Extender, Data Comm Test Equipment

33. Spectron, 344 New Albany Road, P.O. Box 620, Moorstown, N.J. 08057, (609) 234-5700 Cabinets, Patching Equipment and Cables

34. Racal Vadic, 222 Caspian Drive, Sunnyvale, CA 94086, (408) 744-0810 Modems, Multiplexors, Autodialers

Jim Dowling has been a member of the HP3000 user community since 1976. He co-founded and acted as the first chairman of the New England Regional Users Group and has served on the HPIUG Contributed Software Library Committee since 1978, the past two years as chairman. Jim has been with Bose Corporation since 1969 working in Engineering, Project Management, Administrative Systems and since 1978 in Data Processing as Operations Manager. Mathematics, Physics and Electronics Engineering form his fields of Academic training. Computer Science and Information Systems Technology have been a "professional hobby" since he discovered the "magic" of MRP while coordinating the costing and procurement of materials for Bose' largest government contract (a nuclear reactor con- trol system, believe it or not). His two most exciting discoveries in D/P have been Recursive Programming and Data Dictionaries, the former because it taught a lesson on the dangers of applying a tool improperly just because it was an elegant technique and the latter because it prom- ises to be the key to developing Information Technology tools just as the micro-processor has been the key to realizing computing tools.

Managing and Supporting Personal Computers In a Datacenter Environment

by Jeffrey J. Blau

In the last ten years, there has been tremendous growth in the data processing industry. With this growth, many technological advances have occurred. No area of data processing has shown faster technological advances, greater growth, or more future potential than in the area of personal computers. Bacause of such growth and technological advances, managing and supporting personal computers in large corporations has become extremely important. Du Pont, as the largest chemical company in the world, has put in place a strategy and organization to handle personal computers. This paper will describe what Du Pont is doing to manage and support personal computers.

DEFINING A PERSONAL COMPUTER

Personal computers go by many names; P.C.'s, micros, micro-computers, desktop computers, workstations to name the more popular. At Du Pont presently, personal computers must have a processing unit, display monitor, mass storage, printer access and telecommunications capability. In a more general sense, a personal computer fits on a desk or small table including most peripherals, has memory capacity of 64Kb to 1Mb+ and is priced under $10,000. The price must be low enough to justify use by an individual or for a single application. We call them personal computers because the 'personal' aspect is important. The emphasis is the one-to-one personal interaction between worker and computer to solve his/her particular problem.

THE DU PONT COMPANY - DIVERSITY AND AUTONOMY

The Du Pont Company, excluding Conoco (the ninth largest U.S. oil company), is the largest chemical company in the world. Du Pont makes a very wide range of products from chemicals to pharmaceuticals to fibers to laboratory equipment to electronic products. The company is organized into nine industrial and fourteen staff departments. The industrial departments are the revenue producers while the staff departments supply cost-effective support services to the others. Each industrial department is much like a semi-autonomous company of its own, running itself like an independent entity. Because of this autonomy, central support and management in certain areas can sometimes be difficult for the staff departments. Personal computer management and support represents such a situation, being a responsibility of the Information Systems Department, a staff department.

FORCES DRIVING PERSONAL COMPUTER USE AT DU PONT

Personal computer use is becoming important at Du Pont. Several forces are driving this use:

* People want control of their own work - People do not want to depend on others to get their work done. The primary dependence here is upon large mainframe computer systems. Users have little or no

control over resources available, end-user tools, turnaround time, or costs.

* Competition requires faster business decisions - Decision makers need to have the ability to get information quickly in a usable form.

* Production data processing vs. decision support systems - While the need for regularly scheduled work on large mainframes continues, the need for interactive decision support systems is growing quickly. Most mainframe systems cannot handle these interactive requests as well as mini's or personal computers.

* Application development backlog - Du Pont, like most companies, has more work requests than there are skilled personnel to handle them. In addition, with tighter budgets, cost justification for many useful projects becomes difficult.

* Availability of 'user-friendly' software packages - Most of the software now being produced is extremely easy to use. This allows the user to handle his/her own computer requests without the need or expense of using a skilled data processing professional.

* Hardware technology changing rapidly - Every time one picks up a computer industry publication, there is another announcement for a new or better personal computer. Most times there is more functionality and power for less money. It is hard to pass up the tremendous capabilities personal computers have for the prices asked.

* Purchase authorization authority moving to lower management levels - Because of the low price of personal computers, lower level management has the authority to purchase them. This has led to a demand by lower management to acquire their own computing capability.

PROBLEMS AND CONCERNS

Along with the forces driving personal computer use and the potential competitive advantages of personal computers versus larger multi-user computers, some problems and concerns arise. Listed below are some of them.

* Many vendors - There are so many different vendors offering personal computers. Each has its advantages and disadvantages. If each user goes out and buys whatever personal computer he/she thinks they need, soon there will be a tremendous proliferation of incompatible computers.

* Rapid technology changes - Personal computer technology is advancing rapidly. This is making present personal computers obsolete very quickly. Users must justify their personal computer purchases with a maximum twelve month payback.

* Vendor promises - There is a bad track record by vendors, both large and small, in hardware and software, of not delivering the product as specified and/or not supplying the product within the time period promised. Both these problems have hurt projects and embarrassed more than one person who took the vendor's word.

* Duplication of work - Because of their low price and easy installation and use, many people have bought personal computers and software and then loaded up their application(s). What this has led to is that many people have individually evaluated hardware and software, done their own self-training, and solved their own problems. This is alright in an individual case, but with many people duplicating the work others have done, time, money and effort are wasted.

* Data integrity and security - There is a concern whether the data being used by any person on a personal computer is correct. How is the data obtained, is it being accurately inputted and is it up-to-date are serious data integrity questions. Regarding data security, how safe is confidential data on a personal computer? The only data security advice we can give our personal computer users is to put your confidential data on floppy disks and lock them up. If you have a hard disk system you can buy a locking computer cabinet. Other than that, data security for personal computers has not been seriously addressed yet.

MANAGEMENT AND SUPPORT

In light of what has been presented so far; that is, Du Pont diversity and autonomy, forces driving personal computer use, and problems and concerns with personal computers, it is easy to see why a program to manage and support personal computers is a must. In particular, we must manage

* network communications,

* standards of good practice,

* integration of packages,
* user education,
* data security, and
* vendor and corporate support.

In order to manage and support these areas, the Information Systems Our mission states that our group will "provide corporate leadership in personal computing/workstation technologies, monitor corporate Du Pont user requirements, promote compatibility among personal computers and with other distributed technologies, facilitate hardware and software selection and acquisition." To fulfill this mission there are three major areas in which we deal;

* guidelines,
* central user support, and
* maintain technical currency of staff.

GUIDELINES

The first major area to facilitate managing and supporting personal computers is the use of written guidelines. These guidelines help the prospective personal computer purchaser decide which machine and software he/she should buy. The benefits of selecting from the guidelines are that with reasonable assurances

* the system has already been evaluated,
* experienced help is more likely to be available,
* communications capability in the Du Pont environment has been evaluated, and
* adequacy of software libraries has been determined.

Below are some of the important aspects of the guidelines.

* Hardware - Personal computers may be acquired from any of Du Pont's primary hardware vendors - IBM, Hewlett-Packard (HP) , or Digital Equipment (DEC). These vendors are also the only ones guidelined for mini and mainframe computers. Within these vendors, ISD will support selected models that are believed to have broad business applicability and are consistent with future directions for that vendor and the industry in general. The models are selected after evaluation by the Personal Computer Technology Group. Any location proposing to purchase personal computers manufactured by vendors outside the guidelines must submit a proposal for review to ISD's Planning Division stating why no guidelined vendor's products will meet their needs. Third party hardware such as printers, monitors, and circuit boards may be acquired when not available from the primary guidelined vendor.
* Software - The Personal Computer Technology Group has put together a matrix of software which the group will support. The matrix consists of the selected models of the guidelined vendors on one axis and categories of software on the other axis. These categories include
  - operating systems,
  - language(s),
  - spreadsheets,
  - data management,
  - word processing,
  - file editing,
  - communications,
  - graphics, and
  - integrated packages.

  There may be one or more packages for a selected model for a particular category or there may be none. The user is not limited to these packages, but ISD support may not be available. Use of personal computer application packages is encouraged; thus the need to develop unique software packages is probably unnecessary.

Because personal computer technology is changing so rapidly, it is essential that proposals reflect a quick financial payback (one year or less is recommended). Expect to reevaluate personal computer needs for the next application. Equipment chosen today is likely to be obsolete in less than two years.

In order for these guidelines to work, they must be well managed. Guideline management has four aspects.

* Acceptance - The guidelines must be accepted by the departmental data processing managers. In addition, these guidelines must also be accepted by the users.
* Control - If someone wants to buy a personal computer from a non-guidelined vendor, they must submit for approval a

document called an Information Systems Proposal. A second control is done by Du Pont purchasing. Most personal computer purchases are done by a central purchasing group. This group will question any attempt to purchase non-guidelined equipment. The third control is the internal auditors.

* Deviations - Economics should not be the sole basis for choice of personal computers. Many of the proposals for non-guidelined personal computers are based on buying less expensive machines. Other factors are often overlooked which can outweigh the price difference. If viable guidelined alternatives are available, they should be chosen. If no guidelined alternative is available, conditional approval will be given for the one project only.

* Evaluate promising candidates outside guidelines - Our technology group continues to evaluate other equipment. The guidelines must not become static or they will soon be obsolete. Serious consideration will be given any equipment which has a significant advantage in terms of functionality. Portables and transportables are items under consideration now.

CENTRAL USER SUPPORT

Central user support is one of the primary responsibilities of the Personal Computer Technology Group. Activities included in this area are

* personal computer training,
* 'hands-on' demonstrations,
* consulting,
* Personal Computer Newsletter,
* personal computer bulletin board, and
* organizing and chairing the Departmental Personal Computer

  Coordinators Committee.

More detail will be given later about training and the Coordinators Committee.

* 'Hands-on' demonstrations - The technology group has at least one of every guidelined personal computer. We often give 'hands-on' demonstrations of these computers to any Du Ponter who needs to know more about a certain computer before making a purchase decision. We also give introductroy demos to people who have never used a personal computer. We are also organizing a special demonstration room where Du Ponters can sign up to use or test on a guidelined personal computer.

* Consulting - Consulting consists mostly of answering routine questions over the telephone and showing prospective computer users how personal computers might be one alternative solution to their particular need.

* Personal Computer Newsletter - The newsletter was published quarterly in 1983, but we hope to publish it every 4 to 6 weeks in 1984. In the newsletter, with a circulation over 1,000, are short articles about new hardware or software and examples of how some people are using personal computers.

* Bulletin board - The bulletin board is presently being implemented. We plan to put on it any late breaking personal computer news such as product announcements or professional seminars along with answers to individuals' questions which are pertinent to all users.

Training

With any relatively new technology, training is extremely important. People in the corporation, from senior management on down must understand the technology to decide whether to acquire it and then how to effectively utilize it. This is very true for personal computers. There is so much in the press these days about personal computers (Time's 1982 "Man of the Year"), yet most people know very little about them. At Du Pont we have a four level approach to training.

* Senior management - For senior management a computer management program has been developed. The goals of this program are 1) to help them become more comfortable with computer technology and applications, 2) to help them to make more effective use of computers in business, and 3) to prepare management to deal with the future of computers in business. This program takes 4 to 5 days. The core topics are

  - computer concepts,
  - stategic issues,
  - planning,

- managing applications,
- impact on people, and
- data management.

Around these core topics the following are discussed:

- business data processing,
- office automation,
- end-user computing,
- technical and scientific computing,
- networking and telecommunications, and
- process automation and control.

For personal computers, there is a 'hands-on' session. During the session personal computers are introduced, a public data bank is accessed, spreadsheets are introduced, information is organized and analyzed, and 'what-if' analysis is done using the public data bank information. The personal computers are also available during any of the free periods. The primary goal is to have the personal computer be viewed as an analog of computer technology.

* Middle management - For middle management a seminar is given to those who will have some direct involvement with data processing. The seminar is an overview of Du Pont's data processing activities. These activities include
  - planning,
  - data center operation,
  - distributed processing,
  - security,
  - database,
  - project management,
  - software engineering,
  - support,
  - personal computers,
  - end-user software, and
  - office automation.

  The purpose is to familiarize middle management with computer concepts and give them a working understanding of specific ISD activities. Personal computer topics include
  - terminology and characteristics,
  - business concerns,
  - applications,
  - personal computer concerns,
  - guidelines,
  - ISD strategy and planning,
  - ISD support,
  - future, and
  - warnings.

* Personal computer coordinators - For the personal computer cooordinators, the training is more technical and more specific to the machines themselves. Training sessions show the coordinators how to replace accessory boards and reconfigure systems to add peripherals, how to hook up various peripherals to the computers, how to run various communciation and application packages, and give them information to help their respective departments make good computer system choices.

* Professionals - For the professional level, several courses are offered on operating personal computers and effectively using various application packages. Members of the technology group will give training classes on request. ISD has a Training Institute which coordinates course scheduling. It has contracted with an outside firm which offers many personal computer courses. There the professionals get classroom training and 'hands-on' experience. The technology group has also written some 'how-to' manuals which are distributed on request. The final area of training for the professionals is the personal computer tutorials. The technology group evaluates tutorials and will tell people which are the best. Du Pont's central Technical Library, on the technical group's recommendation, purchased several copies of a particular tutorial which any Du Ponter can sign out.

To summarize the training effort, there is a certain goal for each level. For senior and middle management, the goal is literacy and acceptance. For the personal computer coordinator, the goal is expertise. For the professional, the goal is productivity.

Departmental Personal Computer Coordinators Committee

Because of the diversity and autonomy of the departments in Du Pont and because of the difficulty of one small group trying to support the entire company, the Personal Computer Technology Group has organized and now chairs a committee called the Departmental Personal Computer Coordinators Committee. The committee is composed of a representative or representatives of each department, assigned by the department's data processing manager. The functions of the coordinators are

* to be the central reference point for their respective departments,
* to handle first-level problem determination,
* to be the departmental personal computer expert, and
* to help formulate and carry out the strategic personal computer direction for their department.

Several activities are scheduled to help the coordinators do these functions. First there are monthly meetings. At these meetings, presentations are made about applications people are doing, issues that are important to personal computers and/or Du Pont policy, guidelined vendor hardware, and new software packages. At this meeting and at a special session between meetings, question and answer periods are held. Anyone can ask or answer a question concerning personal computers. Another coordinator help is the bulletin board, which will contain any important items for the coordinators. Vendor sessions are also scheduled to discuss new product announcements and vendor personal computer direction. Finally, the technology group does consultations with individual coordinators to help them with specific items.

This committee is an excellent tool to help manage and support the personal computer effort. One of its most important advantages is communications. Every department has at least one representative so all the departments should know what is going on with personal computers. With the meetings, presentations, and question and answer sessions, lots of information is exchanged. This information includes company guidelines, hardware and software evaluation and corporate strategy and direction. This exchange is also beneficial to the technical group. The coordinators are both enthusiastic and cooperative, creating a positive and productive organization. In terms of support, the coordinators are able to handle quickly many of the routine questions which departmental users have. Finally, by being able to talk to such a representative group, the departments will all know the overall corporate direction for personal computers.

MAINTAIN TECHNICAL CURRENCY OF STAFF

The last area which the Personal Computer Technology Group deals with in managing and supporting personal computers is maintaining technical currency. What this means is keeping up with all the industry advances. Activities to accomplish this include

* reading and reviewing the current literature,
* evaluating new hardware and software,
* attending seminars and conferences, and
* having strategic direction sessions with our guidelined vendors.

No one can effectively manage and support a particular technology for a company unless they are constantly aware of what is happening, what is available, how it works, and where the technology is headed.

One particular activity in this area is a workstation prototype project. The technology group is building prototype workstations from our three guidleined hardware vendors. The goal is to take a guidelined personal computer, integrate as many functions as possible, and then let people evaluate it. Integration can be one package that can do spreadsheets, graphics, database, etc. or a set of packages that can use each others files easily. Integration also includes integrating with an office automation system on a mini or mainframe with a minimum electronic mail connection. This project has taught us a lot about package integration, electronic mail, personal computer to host communication and user requirements for a workstation.

CONCLUSION

This paper has described personal computer management and support at Du Pont. The first part explained the importance of managing and supporting personal computers to the company. The three major areas of management and support; guidelines, central user support, and maintaining technical currency of staff were

explained and activities relating to each area were described. While it cannot be denied that many individuals within the company would have found productive uses for personal computers without all that has been described, these activities have brought personal computers into the company in a more orderly manner, made many more users productive quickly, and set a fairly stable course in a rather turbulent technology.

Biographical Sketch

Name: *Jeffrey J. Blau*

Title: *Senior Systems Analyst*

Company: *E. I. Du Pont de Nemours and Company, Inc.*

Address: *Information Systems Department G-560 Wilmington, DE 19898*

Telephone: *(302) 774-3725*

Jeff Blau has been in data processing for about 12 years, the last 11 with the Du Pont Company. He has a B.S. and M.B.A. from the University of Delaware. His first 5 years were spent doing programming/analysis work on large IBM main-frames using DOS, 370 OS, and MVS. He did system programming/operating system support in MVS/JES3 for the next 2 years. This was followed with a 2 year assignment as a computer operations supervisor in Du Pont's central computer center. After this, he was a systems manager and systems analyst on a HP 3000. This included being in charge of getting data communications operating among the International Department's world-wide HP 3000 sites. The last year was divided between working on a DEC VAX project and his present assignment with the Personal Computer Technology Group.

IMAGE Locking: Practices and Pitfalls

by Michael A. Casteel
Vice President
Computing Capabilities Corporation
Mountain View, California, USA

The HP3000 literature includes a number of contributions which address the design and implementation of IMAGE data bases and applications using them (see [1] and its bibliography). Unfortunately, with the notable exception of [2], these contributions give short shrift to one of the most critical aspects of data base design and programming: the correct application of IMAGE locking. While this is not such an important consideration for single-user systems or for large batch, "byte-banging" applications, a properly designed and implemented locking strategy is crucial for modern on-line systems.

If a suitable locking strategy is not included in the data base design, on-line performance can suffer due to locking conflicts between users. More importantly, improper design or programming of a locking strategy directly affects the reliability and correctness of the on-line system.

This article describes two common approaches to locking and reveals some hidden flaws in major implementations. As we shall see, it is not only application programmers who may err in their locking strategies: even some commercial transaction processors can damage your data base due to flaws in their application of locking.

Locking-related "bugs" are among the most insidious, since all the programs can appear to work properly but the data base on occasion becomes logically inconsistent. These bugs typically show up when two users try to update the same record at the same time, with the result that only one of the updates is actually recorded in the data base. Both users think their transactions were completed correctly, but at least part of one has been lost.

For a simple example, consider an on-line parts inventory system. Suppose that two users simultaneously issue the same part from inventory. For each user, the program first reads the current inventory record showing, say, a stock on hand of 25, and displays it on the screen. When the first user completes the transaction, the program locks and updates the record, subtracting one part issued from 25, leaving 24. A short time later, as the second user completes the transaction, the program again locks and updates the same record, again subtracts one from 25, leaving 24. As a result, the next physical inventory will show a "shrinkage" of one part due solely to improper locking. While two parts were actually issued, the computer only subtracted one.

Weak Locking

This is an example of what I call "weak" locking, where locks are only placed just prior to the update. In particular, locks are not held during interactions with the user. The relatively short duration of these locks tends to minimize the possibility of contention between users. However, this example raises the more serious question of how to guarantee data base integrity using weak locking. While there is nothing inherently wrong with this approach, the evidence shows that it is very easy to make a mistake (see below). What is required to do it right?

Referring to the simple example above, the program made the fatal assumption that the inventory record being updated had not changed since it was read. The problem here is that IMAGE only guarantees the record will not change AFTER you lock it. In this example, the first user coincidentally changed the record after it was read, but before it was locked by the second user. The probability of such an occurrence depends on the frequency with which different users access the same part, and the length of time between reading the record and locking it. Naturally, this probability is near zero during testing!

For correct operation with weak locking, follow this procedure:

1) Read the record(s) which you will be updating. Show them to the user, think about them, calculate with them, etc. When ready to update, proceed to step 2.

2) Lock all the record(s) which you will be updating.

3) Re-read the record(s). You may use mode 1 (re-read) to minimize overhead, with the risk of an occasional transaction cancellation (see below) due to migrating secondaries on manual masters. This risk won't be high if your data set capacities are adequate.

4) Compare each record you re-read with the contents originally read in step 1. THIS STEP IS ESSENTIAL. You must make sure, one, that the record you originally read is still there, and, two, that it hasn't changed. If the record has been changed or deleted (note that the re-read in step 3 may fail in this case), cancel the transaction. Your thoughts and calculations in step 1 were based on obsolete data.

5) Once that you are assured that a record is still the same, you may update or delete it. If you haven't yet re-read and compared all the records, be prepared to undo (back out) this update/delete if you subsequently find that the transaction must be cancelled.

6) After updating all the records, unlock them.

If you apply these steps to the inventory update example, you will see that they would have prevented the problem. In this example, step 4 would have detected the first user's update and cancelled the second user's update before damaging the data base. But this is not the only pitfall to avoid when using weak locking. The other steps in this procedure also have their purpose. By precisely following the steps, your applications should work correctly. Apparently slight variations can invalidate the whole thing. For example:

1) Not bothering to re-read the records. This gives you no assurance that the original record is still there, much less unchanged. Some other user may have deleted the record, and another user added a completely unrelated record in the same slot where your record used to be.

   This error appears to be made by the TRANSACT processor, as described in [3]. TRANSACT is the transaction processing component of Hewlett Packard's RAPID/3000 package. Although it has been criticized for its use of data base level locking [4], there have been no warnings of this fundamental omission in its locking implementation. Once the problem has been recognized, however, it is possible for the programmer to implement correct locking by coding explicit calls to DBLOCK.

2) Not bothering to compare the record you re-read with the original. Once again, you have no assurance that the record you re-read has anything in common with the original record.

   This step was left out of item 6 under "Database Programming" in [1]. The erroneous inventory count in the above example is one possible consequence of this error. Perhaps more surprising is the possibility of updating or deleting the wrong record. Suppose in our example that the first user actually deleted the record instead of updating it. After the record has been deleted, a third user may add a completely different record in the same slot. When the second user re-reads the record to update it, he reads and updates this new record instead. Now things will really be fouled up!

3) Not keeping the entire original contents for comparison. As this can add up to a lot of memory when many records are to be updated, you might try to compress the record down to a few words, such as a hash total. However, there is a risk with this approach: Because each hash total value equates to a very large number of record values, changes can be made to the record which the hash totaling scheme will fail to detect.

   This method appears to be used in the QUICK processor, as described in [5]. QUICK is the on-line transaction processor in Quasar Systems' POWERHOUSE family. There seems to be no simple way for the programmer to correct this problem within the QUICK environment.

   The degree of risk in this approach depends on how well the hash totaling scheme matches the changes being made. For example, a simple hash totaling scheme might be to take the algebraic sum of the contents of each word in the record. This scheme would be foiled by a typical inventory record update which adds one to Quantity Issued at the same time it subtracts one from Quantity On Hand, with the result that the (algebraic) hash total would not change!

A variation which will work is to keep and compare the original contents of only those items to be updated. But be very careful not to change any other items; another user may have been using or changing them.

4) Only locking, re-reading, updating and unlocking one record at a time can also be a problem. Several records could be updated before discovering one that has been changed or deleted. The transaction must then be cancelled and all the updates undone. But if one of the records that was updated and then unlocked has since been updated by another user, a logically inconsistent data base results. In order to cancel your transaction you must back out the other user's update. If you don't, you haven't completely cancelled your transaction.

   This also appears to be the case in QUICK [5].

As you can see, it is easy to make a mistake in programming weak locking; a lot of people have. While there are variations which work correctly, any mistake places your system at risk of creating a logically inconsistent data base. The four errors listed here might only damage your data base infrequently, but you would naturally prefer not to leave such holes in the logic of your system.

Strong Locking

An alternative to weak locking is what I call "strong" locking, wherein all the records to be updated are locked before they are read, and not unlocked until after they have been updated. This approach has the advantage of being simpler and less error-prone. The original procedure becomes:

1) Lock the record(s) which you will be updating.

2) Read the record(s) which you will be updating. Show them to the user, think about them, calculate with them, etc. When you are ready to update, proceed to step 3.

3) You know that the records you originally read haven't been changed, deleted or moved, since you have kept them locked. Just go ahead and do your updates/deletes, and unlock when you are done.

While this method is clearly easier to implement and maintain, thus promising more reliable software, it is not without its price. The use of strong locking often involves holding locks while interacting with the user, possibly for long periods. This raises the possibility that conflicting locks could delay one user while another finishes a transaction. On the plus side, there is less risk of damage to your data base due to subtle errors in implementation. At the same time, strong locking also reduces overhead by eliminating the need to re-read and compare every record before updating.

The usual practice when using strong locking is to minimize the likelihood of conflicting locks by using entry level locks, with a judicious choice of the data item to be used for locking each data set (see [2] and below). Using entry level locks with strong locking usually works quite well for all but a few complex transactions, provided that the data base and locking strategy have been carefully designed.

Choosing a Locking Strategy

The choice between weak and strong locking methods is not the only decision to be made in regard to locking. There is also the question of the detailed locking strategy, i.e. which levels of locking to use: data base, data set, or entry level?

At one time, data base locking was the only option IMAGE offered. Even with the relatively short duration of weak locks, using data base locking will limit throughput by allowing only one user to be performing updates at any time. Data set locking is better, where the user only locks the data sets which are to be updated. While this allows other users to be updating other data sets, it still delays users who wish to update other records in any of the locked data sets.

The option with the least likelihood of delaying users is entry level locking. In IMAGE, this is the logical equivalent of record locking. Rather than locking specified records, however, IMAGE locks any records within a particular data set which contain certain data item values. For example, you can lock records with PART-NO equal to 312, or DATE-ADDED equal to December 31, 1982. It is also possible to lock using other comparison relations, such as DATE-ADDED greater than December 31, 1982.

With entry level locking, no user need wait for a lock unless another user holds a similar lock, e.g. with the same PART-NO, or an incompatible lock. An entry level lock is incompatible with a data base or data set lock, or with an entry level lock on the same data set using a different data item. If one user is delayed by an incompatible lock, subsequent users will also be delayed, even if their lock requests are compatible with locks already held.

IMAGE manual master data sets often pose such locking conflicts. IMAGE will not add or

delete records in manual master data sets without a data set lock. This restriction conflicts with the user of entry level locking.

For example, one user locks a record while entering changes to a manual master. When the next user tries to add a record to that data set, IMAGE requires that the entire data set be locked. Since the first user has a record locked, IMAGE makes the second user wait until the first one is finished. All subsequent users also have to wait, even if they only want to lock a record. As a result, the entire data set is "locked up" until the first user unlocks the one record. This result is the same when different programs use different items for entry level locks.

If on-line adds and deletes to the manual master are infrequent, this may not be a problem. On the other hand, if your data base design includes manual masters which you expect to be subject to a significant number of on-line adds, deletes and updates, then you may wish to consider making it a detail data set instead.

This shows how important it is to consider locking strategy during the design of the data base. You can minimize locking delays by using entry level locking choosing a single item in each data set to be used as the lock item. (This is usually a search item, although it need not be.) It is very important that all programs use the same data item for locking. If any program (or transaction) uses a different item, the resulting conflicts in locking can be as bad as when using data set level locks.

In a purchasing system, for example, data sets containing order headers, order lines and receiving transactions could all be locked using the Purchase Order Number data item. Since it is unlikely that two users would be dealing with the same order simultaneously, there is little risk of delay due to lock contention.

Now, consider the likely on-line usage: If you are using strong locking, will the program know the data item value in order to place the lock before reading the record? If not, it will have to read the record, place the lock, then read it again, just as in weak locking.

For example, suppose that the receiving transaction in our purchasing system retrieves order lines by part number without knowing the purchase order number. The part number must not be used for locking; this would conflict with locking on purchase order number. Having retrieved the proper order line record, place the lock using the purchase order number from the record. Re-read the record and test to make sure that it's still the proper record. Remember that it may have been changed (or even deleted) since it was read before locking.

Finally, is it likely that more than one user at a time will attempt to access and update records in the same data set with the same value for this lock item? If so, your users could experience delays due to lock contention. You may wish to consider using a different data item for locking.

Combining Strong and Weak Locking

As mentioned earlier, there are some complex transactions which do not lend themselves to strong locking. Perhaps every user is expected to update a single control record in this transaction. Keeping this record locked for any length of time will surely delay other users.

In cases like this you may wish to combine the two types of locking in a given application: use strong locking where possible, for simplicity and reliability; use weak locking on transactions where you need short lock duration. This will work fine, PROVIDED that you:

1) Define and use an appropriate entry level locking strategy. This is important in order to avoid delaying weak locking transactions at long-held strong locks.

2) Correctly program your weak locks.

In the example where many users need to update a single control record, strong locking is fine for most of the transaction, to hold locks over all the other (data) records to be updated. Then, when it's time to update, lock, re-read, update and unlock the control record. But be careful to avoid the following pitfalls:

First of all, placing the control record lock while other records are already locked will require that your programs possess MR (Multiple RIN) capability. Don't take this capability lightly, as the risk of deadlocking your application is very real. In this example, you must make sure that no one who already has the control record locked tries to lock a data record. This will deadlock with a user who has that data record locked while waiting for the control record.

Finally, IMAGE doesn't allow you to unlock just the control record; when you unlock, all your strong locks will be removed as well. This is fine if you have updated all the data records as well as the control record, just before unlocking everything. If you need to keep your other locks in order to continue the transaction, then you must place the control record in another data base, or access it using a second DBOPEN.

Locking and MPE/KSAM Files

These locking principles apply to MPE files as well as to IMAGE data bases. Since the MPE file system does not provide record locking, it is harder to use strong locking without encountering locking conflicts among users. Even weak locking is more complicated, since locking all the files to be updated (in order to avoid weak locking problem (4) above) requires the use of MR capability, with special care to avoid deadlock.

One scheme to support strong locking when using MPE files is to create a dummy IMAGE data set corresponding to each MPE file, and use entry level locking on the dummy data set to effect record locking. This scheme is used in the INSIGHT transaction processor from Computing Capabilities Corporation, as described in [6].

MR capability must still be used in order to lock each file around the actual update (FLOCK / FUPDATE / FUNLOCK) while holding the IMAGE entry locks. With KSAM files, you should re-read by key to reposition in the file before update. Note: This action is advised in the KSAM manual, but I have not yet observed the situations in which it is actually needed. No doubt these situations involve restructuring of the KSAM index.

Conclusion

Although it is essential to the correct operation of multi-user on-line systems, the subject of locking strategy and implementation has been largely ignored in the literature. It is in fact not a simple subject. This article has described two common approaches to locking and pointed out subtle flaws in some major implementations.

While the technique of strong locking has been criticized for potential delays due to locking conflicts, it is easier to guarantee data base integrity with strong locking than with weak locking. When combined with the proper use of entry level locking, strong locking need not delay users except when they attempt to update the same record at the same time. In such an event, a proper implementation of weak locking will not delay a user, but is likely to produce a transaction abort instead.

REFERENCES

[1] David J. Greer, "IMAGE/COBOL: Practical Guidelines", Journal of the HPIUG, Vol. 6 No. 1, Jan/Mar 1983.

[2] Gerald W. Davidson, "IMAGE Locking and Application Design", Journal of the HPGSUG, Vol. IV, No. 1, First Quarter 1981.

[3] Hewlett-Packard Company, "TRANSACT/3000 Reference Manual", Second Edition, December 1982, Appendix C (flowcharts).

[4] M.P. Ashdown, "Developing Large Integrated Systems Using RAPID/3000", Proceedings of the HPIUG Conference, Edinburgh, 1983.

[5] Quasar Systems Ltd, "You are the QUICK Screen Designer", Second Edition, December 1982, pp. 184, 186-7.

[6] Computing Capabilities Corporation, "INSIGHT II Reference Manual", Fifth Edition, December 1982, p. 113.

Small World, but how should it be organized?

Chris Spottiswoode
Synergy Computing

A. Naive Real-Time

You are a healthy human being, aware of what is going on round about you. Your nervous system relays its sense-perceptions to your brain fast enough for you to be able to respond to threats or opportunities in your immediate surroundings: you have a 100% real-time information system.

Your computerized inventory control system records every inventory movement as it takes place, so you know exactly when to reorder or move or ship your goods: you have a 100% real-time information system.

Both statements are wrong. They represent what I call "The Naive Real-Time Fallacy", propagated by the optimists of this world. And I am not referring only to computer salesmen: for are we not all optimists!

Where is the fallacy? And how can it be corrected?

Let us start with an extreme counterexample. Our eyesight and our reactions are not fast enough to spot and dodge falling meteorites. "Of course not," you respond, "but there aren't enough falling meteorites to bother me. 99.999...% is good enough for me." And mankind's millions of years in the survival race prove you right.

No country is fast enough and powerful enough to spot and neutralize falling ballistic missiles. No company has a perfect inventory movement recording system or inventory reorder algorithm. "Just a matter of improving technology," you may well retort. And you are almost right. But not quite, for the solution may prove more costly than the problem, and finally our competitors can improve too. We should not even expect anywhere near 100% real-timeness or accuracy. "Just enough!" should be our watchword, as indeed it has always been throughout the saga of evolution.

So let us accept that we have 90% real-time information systems, which are quite good enough for us. Our knowledge of what matters in the world around us marches along with that world, even though we are sometimes a little behind in our perceptions or reactions.

Many first-time purchasers of inventory control systems are still at this stage of understanding of the system they are expecting. Regrettably many designers of real-time inventory control systems are still there too! For this is still Naive Real-Time. The above qualifications are still not the point: the message from Distributed Processing to Naive Real-Time is not that slow or irregular or infrequent communications make real-time difficult. Let us summarize the naive conception of real-time and see what the real problem is.

Imagine a data-structure diagram of our naive inventory control system: we probably have at least an item master, an inventory master, an inventory movement transaction detail file and various order files, all adding into summary fields for that instant access to the current status. There are data access paths where necessary, and we (the end-user or the programmer) can wander around these paths and establish the exact inventory position, order-status, etc.

As Charlie Bachman ("The Father of Data-Base") puts it (in his ACM Turing Award Lecture of 1973), we are now "navigators in a data-base". The Data-Base Revolution in programming is like the Copernican Revolution in astronomy. The CPU, like the sun, is no longer the center of our universe: we must situate ourselves in a universe of data.

In this Information Age we must take this view, but the mistake one then naively makes is to look at a business enterprise as if it were a human being. A rather similar parallel was drawn in the seventeenth century by Thomas Hobbes, in his book Leviathan, where he compares the state or "commonwealth" with a giant body, with its lines of authority and central sovereign (or as we would say now: lines of communication and central co-ordinating organ). Nowadays we look back on Hobbes as one of the intellectual fathers of political absolutism. This leads us on to look at how the real world has forced us to adopt more decentralized and democratic views.

We shall see that the fundamental error in Naive Real-Time is to assume just one observer or central actor, who sees the entire data-universe as if at the same time, that is, with no time component, no time dimension within which to distinguish the points in the data-space. We shall see the same problem that Einstein saw in the Newtonian universe: there is no simple definition of simultaneity. Naive Real-Time is equivalent to absolute Newtonian space, the whole of it being observed as if at one time. We need to introduce the concept of relativity to Real-Time.

B. The Real World

Naive Real-Time had its heyday in the late sixties, but the centralized on-line mainframe was soon recognized as the monster or Leviathan that it was. The price and limitations of all those telephone lines for so-called instant access, and the complexity of the programming on those big beasts simply made the costs too high. These issues totally obscured the more fundamental design complexities to which we return later, but they were sufficient to lead to rebellion.

Distributed Processing was rising in the seventies even as Charlie Bachman was giving his Turing address (though in 1973 we were doing it before the name was invented!). By the end of the seventies "Small is Beautiful" had become the motto. By 1984 the independent PC, that symbol of the democratic revolt, had been blessed by IBM (and how satisfying that we should be celebrating this in George Orwell's dreaded year!).

But what has happened to that elusive ideal of real-time information? With slow or infrequent communications between distributed files, it has some definite limitations. How much has it mattered? Briefly: not much.

Clearly there are major application areas where a high degree of real-timeness is essential and used: airline reservations and automatic bank tellers are familiar examples, though even here off-line or degraded modes, with their various consequences, are part of the total design. But the majority of enterprises with integrated computerized applications simply do not require total 100% (or 90%) real-time systems. Why not?

BECAUSE THERE IS A
FUNDAMENTAL DIFFERENCE
BETWEEN AN INDIVIDUAL
AND AN ORGANIZATION.

Organizations exist because the individual cannot do everything himself. In our complex society we must specialize. We each live and work in our own little worlds, communicating and co-operating with others. Informal communications between the individuals in any one department enable that department to act as one person, with one real-time. An organization consists of multiple departments, each with its own perfectly valid naive real-time.

In a typical manufacturing and distribution company, we do not involve the sales order processing department with the details of the factory order planning, at most we let them know the results. In some applications warehouse staff might not use the theoretical inventory levels at all: it is their job to move goods and record those movements. Frequently the theoretical levels will only coincide with the physical levels at stock-take, since they may at other times refer to unsold inventory, which is not the same as inventory actually in the warehouse.

The sales order processing department is not concerned with the processing of customer billings and payments. That is the job of the Credit Control department, who in their turn update the on-line customer file with credit status, not "real-time" balance, so that orders for risky customers are referred to them.

So we have two examples of "real-time" values, customer credit balance and inventory balance, which do not necessarily refer to any "real" balance in the world represented. Why not? There are two main reasons. The first is the obvious one that the balance may be incorrect

or misleading. Not all relevant movements may have been entered, and others may have been entered incorrectly (This is part of what made us reduce the 100% to 90% real-timeness). The main reason is however that the real-time balance has a different meaning, such as unsold inventory, which refers not to physical inventory but for example to on-hand plus expected from factory plus resaleable returns less expected shippings for some known period. And it is usually quite a task to relate theoretical to physical inventory from time to time.

The meaning of any value on the real-time database depends on the point of view of the user for whom it is intended. Now this is so obvious that it is well to reflect on why it has become necessary to say it. Well: the culprit is the dogma of non-redundancy of data, created by the DBMS missionaries! Before DBMS, each department had its own files, so of course they were looked at from the point of view of that department. Now we are tempted to have one real-time inventory balance field, and it becomes a non-trivial task to relate it to its various potential users.

But relate we must! Without further ado, let me describe the Relativistic Real-Time Data-Universe.

Modern organizations are not run by some superhuman Leviathan or Big Brother who knows everything in (naive) real-time and directs activities with infallibility. An organization, by definition, consists of separate departments organized to achieve some more or less precisely understood objective. Each department runs in its own real-time, isolated to a greater or lesser degree from the others. Our limited minds require that we work in simplified worlds, and historically organizations have evolved taking the sizes of our minds into account. (As an aside, one wonders how the much-trumpeted Fifth Generation will cope with the essentially human problem of assessing what people can take.)

As in astronomy, all points of view can start by defining themselves relative to the fixed stars, which may in our case be an Item Master and a Customer Master, though at this point without such attributes as balances.

Then a Credit Control department, for example, will add balances, credit status, transaction entry files with associated controls, and sundry other local data. They will work independently of the order-processing department, say, except in two respects. The first involves the Accounts Receivable statement close, when Credit Control determines that its transaction data is complete and accurate enough, and initiates the close. The balances are updated and the customer credit statuses are deduced as a function of the balances and other factors. Order-processing might refuse orders or refer them to Credit Control as determined by these credit statuses, which are derived from the same transactions that are communicated to the customers on their statements. These referrals of orders back to Credit Control represent the second type of interaction between our two departments. Let us characterize these two types of interaction in more detail.

The second (we return to the first later) was a stream of referrals of incoming orders back to Credit Control for possible credit authorization or other follow-up. This is a queue (or queues) of objects or entities representing a flow of units of work between the two departments. There will be more such queues, such as the queue of authorized orders passing on to the shipping department.

The first type of inter-departmental interaction was the close, in the above case an Accounts Receivable statement close. Now we associate this with a batch job on the computer and hence tend to think of it as a computer-created evil. Not so. Such computer batch jobs are always associated with a real application event: a cut-off. In this case Credit Control triggers the event by "cutting off" the A/R transaction flow from outside the computer and thereby synchronizes various parties, for example data capture, order-processing, and the customer base. Refusals by an order clerk may be by referring the customer to the state of affairs as it was at this "sync-point". A later "real-time balance" may confuse or mislead any or all of the parties concerned: because of delays and/or errors in payments, mailing, encoding, data-capture, or data-control (to name a few possible sources) a balance at any time other than at a close should perhaps most diplomatically be used by the Credit Control department itself rather than by an order-clerk. The close- or cut-off- related sync-point provides a much more clear and unambiguous communication.

A more interesting application-defined cut-off is to be seen in the physical inventory count or stock-take. Here warehouse, shipping, Credit Control (re returns), and accounting have a familiar sync-point around which they orient themselves relative to each other. Inventory movements are frozen, or cut off, during the stock-take. A batch run will "transform" the real-time inventory balance used by order-processing - by

making adjustments such as adding back allocated orders - to make it coincide with the real physical balance. If this balance was being maintained then only the next step might be necessary: an exception report of variances to assist all relevant departments with the error correction or reconciliation process. At other times the separate departments will operate much more independently.

So we build up the picture of each department with its private data, its communications by means of queues, and the synchronization of its own real-time with the real-times of other departments by means of cut-offs and possibly associated batch runs on the computer.

At this point I might just add that these concepts lie behind a package that has now reached an advanced stage of beta testing: SYNQ (for SYNchronizer and Queue-manager) is a system control package based on a system design language called IDIOM, in which the system designer describes the organization in terms appropriate to its own peculiar interrelations.

By means of the SYNQ package, or by appropriate use of, for example, Data-Flow Diagrams, or simply by otherwise emulating SYNQ's queue-management and synchronization functions, the system designer thus defines the interactions between the various departments, whether they are manual or computer-assisted.

We end up with the relativistic picture of an organization's departments each working in their own real-time or time-frame, with the time-frames being related to each other, or synchronized, by transformations effected by means of batch jobs.

This parallelism with Relativity in physics is not merely contrived. Each recognizes that we must start with the viewpoint of the observer in his department or frame of reference; that absolute simultaneity between such viewpoints is not possible, since simultaneity is based on finite-speed limited communication between viewpoints; that different viewpoints, except where they share common fixed entities or resources, can only be made to correspond by means of transformations of data between viewpoints.

C. Consequences for computer system design

C.1 In Application Design

It is safe to design an application for use within any one department. Naive real-time is good enough. If the entire application is managed in detail by one person, then you can obviously ignore Relativity. This is one reason for the success of micros. And PCs will continue to play a major role in the more isolated activities, whether partly on-line or not.

But if your intended computerized application spans departments look for the shared static resources and for the interfaces and transformations - the communicating queues and synchronizing cut-offs. In this way you reduce a complex integrated organization to multiple, simple, end-user understood viewpoints in which Naive Real-Time holds true. If you are cautious you will not change the departmentalization, because it defines the simplified worlds of the actors in your drama.

Incidentally you will find that your database design is much simplified, because you too will be able to work in largely separate simplified worlds or databases.

But if you want to remove or change inter-departmental walls, beware! The move may be justified, but it will only be accepted and work if the computerization removes the raison d'etre for the barriers, and if the effect can be seen by the end-users to be a simplification. My warning is not that change is always resisted: my experience is rather that change is welcomed if it can be perceived to be a simplification. It is confusion and perplexity that is resisted.

Now if the computer can simplify the activities of the organization, how is this to be brought home to the first and all later users? The first must be sold on the benefits. Prototyping and system modelling or simulation would help. Then all users must be able to find out how the new organization hangs together. Remember that the new technological organization probably relies heavily on the computer for its communications. On-line help is probably the answer.

The structure of this on-line help must also be Relativistic: a Leviathan of a text, even one beautifully organized in top-down fashion, will probably not be read. Your on-line help must be structured to mirror the organization:

simple and relative to the context, but it must also be possible to easily establish the role of the current activity in any continuing interdepartmental communication or approaching cut-off.

Finally, the operations of the computerized organization should be under the control of the end-users. Those batch runs are not yours, they are part of the end-users' regular functions and should not be hidden from them. If a particular close normally takes two hours, the end-user wants to know and draw up his schedule accordingly. He must be able to find past run time history and (within limits) program different schedules on the computer.

An end-user oriented application operating system is indicated. You can program your own (Who has not written an applications monitor or control program?) and build in those modelling and on-line help functions, or you could use a package like SYNQ.

C.2 In technical design

One often hears statements such as this: In on-line systems, 75% of the difficulty is in ensuring data integrity. This is surely true if one broadly interprets integrity to encompass access security, data consistency, accuracy, and fault-tolerance.

Clearly a Relativistic approach will result in a rational and simple solution to defining appropriate access rules.

Data consistency is less of a problem the less you are a slave to Naive Real-Time, which dictates that you maintain multiple on-line summaries and access paths, "in case someone wants to use them".

Inaccuracy of data can never be avoided, it can only be reduced. Errors must be controllable, that is, detectable and correctable before too much damage is done. This is one of the major functions of the separation of departments: get the work done quickly, but don't pass it on to anyone else until it is complete and accurate enough. That is what cut-offs are all about.

By fault-tolerance I mean tolerance to technical computer faults such as power or system or program failures. There is no simple solution to these problems, only compromises. But there is one general approach to this set of problems: the system must be restartable or reconfigurable from known checkpoints or sync-points. On-line programs can use logging and recovery mechanisms, batch programs must be broken down into restartable steps. Clearly an application designed with rational and clearly understood interrelationships between functions, that is, a Relativistic design, will more easily be made restartable and have more easily designed and implemented degraded modes.

D. Conclusion

This paper has taken a largely informal and sometimes high-faluting approach to what is essentially a very simple, though much overlooked, issue: some common traps into which the application designer can fall when designing integrated applications.

The human and philosophical aspects were emphasized because they are part of the problem, and the analogy with Relativity concentrates the mind on the essence of the logical problem.

A more formal and systematic formulation can be made (though this cannot be done here) of the simple concepts this analysis reveals. Such an approach - with which we are experimenting at the moment - will create opportunities ranging from Critical Path analyses for integrating inter-department scheduling with the D.P. implementation, to network resolution exercises for simplifying the multi-department organization.

Chris Spottiswoode has been in D.P. for over 14 years, the first 6 with a manufacturing and distribution company, for which he implemented the first mini-based nation-wide commercial distributed processing system in South Africa. Robert Gibson and he formed Synergy Computing in 1978 and in 1979 were responsible for the first purely commercial HP3000 in South Africa. Synergy is the Distributor in S.A. for Robelle Consulting, Quasar Systems, VESOFT, and Account-A-Call. Chris is married, with 3 children, and enjoys mountaineering, skiing, sailing, tennis, philosophy, and politics.

SYSTEM MANAGERS SELF HELP PROCEDURES

by JOHN VEGA/SE

I. KNOWING YOUR SYSTEM .....WHAT MPE VERSION ARE YOU ON

The operating system, MPE, has really been active in the last couple of years. Different versions of MPE were introduced with such vigor that most customers found it very difficult to keep up with the various versions introduced. Issue Number 30 of the communicator finally addresses the situation and suggests a uniform method of identification. Using the convention described by COMPUTER SYSTEMS DIVISION (CSD) I will add a small amendment to the scheme they suggest. Q-MIT is definitely DATE CODE 2244, although the v.uu.ff is slightly different than the C.01.00 suggested. The v(version), uu(update) and ff(fix-level) is changed for your version of Q-MIT to be C.B1.00. This means your version of "Q" has been updated twice (C.A1.00 and C.B1.00). I've included the article by CSD for your reading pleasure.

BENIFITS OF THE "Q" RELEASE

Issue 30 of the COMMUNICATOR gives a brief run down of the enhancements available in the "Q" release (or should I say C.B1.00). Just to hit the high points: Enhanced STORE facility, V/PLUS (VIEW) local forms storage, and several other improvements.

TAPES IMPORTANT TO YOU

As a SYSTEM MANAGER you should keep track of several important tapes. Here is a list of tapes you must safeguard:

OFFICIAL COLD LOAD AND SUBSYSTEMS TAPE....This tape is generated when you are updated to your current level of software. You should rely on this tape to restore any system software. For instance, somehow EDITOR doesn't work as it used to. You would RESTORE the EDITOR file from this tape. You can also use this tape to overlay MPE. By performing a COLDLOAD from TAPE using the UPDATE option only MPE is overlayed. This may be necessary if the current PROPAGATED OFFICIAL COLD LOAD TAPE needs to be recreated.

PROPAGATED OFFICIAL COLDLOAD TAPE....This tape is your current COLD LOAD TAPE. In the event of a SYSTEM INTERRUPTION you would perform a COLDSTART from this tape using the COLD OPTION. Procedures for keeping this tape current are in the section on PROPAGATION.

DISC UTILITY SUBSYSTEM TAPE (DUS)....This tape contains the tools which your CUSTOMER ENGINEER (CE) will use to diagnose system and peripheral problems. This tape will be re-created each time your system is updated to a new revision of MPE SOFTWARE.

II. THE OFFICIAL COLD LOAD TAPE

Each time another set of operating system software becomes available from the factory I will schedule approximately two hours to update your system to the new version. To ensure that the update proceeds properly, I will ask that a certain amount of space is available on disc (typically 10,000 contiguous sectors) and that your system is fully backed up.

The update concludes with the generation of an important tape. We will call this tape the "OFFICIAL COLD LOAD TAPE".

Having created this tape immediately after loading the new operating system we are assured that it contains data of the highest integrity. We can appreciate the importance of knowing software (all those combinations of 1's and 0's) has high integrity when we are confronted with a problem. To be able to tell whether the problem is caused by "HARDWARE" or "SOFTWARE" can be as simple as a "COLD LOAD" from the "OFFICIAL COLD LOAD TAPE". What this cold load establishes is that our software base is as good as it was when I originally performed the update.

The contents of this so called "OFFICIAL TAPE" necessitates some further discussion.. Since we are dealing with several components when we speak of the "OFFICIAL COLD LOAD TAPE" we need to know what the components are.

PROPAGATION of the "OFFICIAL COLD LOAD TAPE"

To propagate means to pass along or continue. If we view the "OFFICIAL COLD LOAD TAPE" we find the components to be: MPE, SYSTEM TABLES, I/O CONFIGURATION, SUBSYSTEM FILES and DIAGNOSTICS. In the development of a computer installation we may have a need to change two of the components which make up the "OFFICIAL COLD LOAD TAPE" (system tables and the I/O configuration). How do we maintain or "PROPAGATE" that all important "OFFICIAL COLD LOAD TAPE" before these changes occur? With minimal effort the propagation can become second nature. Whenever a need arises to change either system tables or the I/O configuration think about "propagation". First, "COLD LOAD" from the "OFFICIAL COLD LOAD TAPE" using the "COLD" option. This will overlay the current MPE, SYSTEM TABLES and I/O CONFIGURATION on disc so that you are assured they're as good as they can be. Now, make your changes to the system tables or I/O configuration via "SYSDUMP".

The tape created by "SYSDUMP" must now be used to "COLD LOAD" the system with the "COLD" option. This brings those changes made during "SYSDUMP" onto the system disc and "INITIAL" builds MPE in memory with those changes.

That same "SYSDUMP TAPE" is now your "PROPAGATED OFFICIAL COLD LOAD TAPE" and should be treated with respect. You would now retire the original "OFFICIAL COLD LOAD TAPE" to the archives. Any future changes to the system tables or I/O configuration would necessitate duplicating this process once again. Only now the "OFFICIAL COLD LOAD TAPE" is the "PROPAGATED OFFICIAL COLD LOAD TAPE".

By using these guidelines you will help HP help you. Without proper propagation you cannot feel confident that the software originally installed is of the high integrity you expect from HEWLETT-PACKARD.

ADDITIONAL MEASURES

Now that we propagate our official cold load tape properly, you might ask why it is necessary to keep the "ORIGINAL OFFICIAL COLD LOAD TAPE". If for some unknown reason your alternate system manager made some changes to either the system tables or the I/O configuration without propagating, or the propagated official cold load tape you normally use to cold start your system mysteriously disappears, what then? Since you have your "OFFICIAL COLD LOAD TAPE" you can start the propagation process all over again by performing a "COLD LOAD" from the "OFFICIAL COLD LOAD TAPE" using the "UPDATE" option. This overlays MPE on disc with MPE from tape but the sytem tables and I/O configuration are not changed. Once the system comes up you can perform a sysdump to generate a "PROPAGATED OFFICIAL COLD LOAD TAPE"

SPECIAL CONSIDERATIONS

As a system manager I would perform one additional step after I am updated to the current revision of the operating system. I would take the "OFFICIAL COLD LOAD TAPE" and archive it immediately. Then, I would generate a "FUTURE DATE SYSDUMP" which would carry the name "OFFICIAL COLD LOAD TAPE" (since I

create this future date sysdump immediately after my system is updated it is identical to the archived version with the exception of not having any subsystem or diagnostic files). Now I have two copies of my current "OFFICIAL COLD LOAD TAPE", just in case. Additionally, I would always keep the "OFFICIAL COLD LOAD TAPE" and the "PROPAGATED OFFICIAL COLD LOAD TAPE" from the previous operating system I was on just for the remote chance that the new operating system didn't perform properly.

If your system implements user procedures in the "SYSTEM SL" the management of the "OFFICIAL COLD LOAD TAPE" is slightly different. The propagation in this case would begin right after I create the "OFFICIAL COLD LOAD TAPE".

USE OF THE "OFFICIAL/PROPAGATED COLD LOAD TAPE"

Whenever there is a possibility of a subsystem being corrupt you would use the "OFFICIAL COLD LOAD TAPE" to restore that subsystem (i.e. COBOL acts strange after a power failure).

Whenever there is a system interruption (power failure, system failure or system hang) you would use the "OFFICIAL COLD LOAD TAPE" or the "PROPOGATED OFFICIAL COLD LOAD TAPE" to "COLD LOAD" your system.

On a periodic basis you would use the "OFFICIAL COLD LOAD TAPE" or the "PROPAGATED OFFICIAL COLD LOAD TAPE" to "COLD LOAD" your system to insure MPE, SYSTEM TABLES and I/O CONFIGURATION have a high degree of integrity.

III. STORE/SYSDUMP TAPE VALIDATION ..... VALIDATE PROGRAM

The whole approach to saving files is something we can never ignore. The use of SYSDUMP to perform this function is one method used by most customers. We all understand that SYSDUMP is merely a STORE with extra information (OPERATING SYSTEM, etc) and in most cases SYSDUMP generates several tapes which we refer to as a backup TAPE SET.

If SYSDUMP completes with no errors one would assume we have a good tape set which could be used if a system reload necessary. This assumption is not totally correct. The HP3000 doesn't check to make sure the data which it sends to the tape drive is in fact what was written. This means the SYSDUMP tape set may not be readable by the system. Of course, if you're in the process of RELOADING your system this situation would be catastrophic.

The possibility of bad data written to tape can be the result of many factors (dirty tapes, poor quality tapes, dirty transport path in the tape drive or hardware failure). Yet, when a tape problem occurs it is typically in the middle of some critical situation (like a RELOAD).

To avoid the frustration (and possible loss of data) one should always verify any STORE which is critical (like a SYSDUMP before a scheduled RELOAD) or any other STORE operation where an error cannot be tolerated. VALIDATE is ideal for this purpose.

VALIDATE will produce a listing of files which are stored on a tape and terminate if an error is encountered. Validate doesn't check the data on tape against the data on disc, but it does check for parity errors, sequence of file marks and size of records which make up the files. These checks and a view of the listing VALIDATE produces for proper file, group and account names should give you confidence that the data on tape can be read by the system.

I recommend VALIDATE be used on any tape set that will be used for RELOAD purposes. If VALIDATE finds an error it will abort or be apparent from the listing.

IV. LOG FILE MANAGEMENT ..... THE LOGERR UTILITY

The idea behind managing log files is to allow the customer to help him/herself. The data stored in the log files is of little use to anyone if the output is difficult to understand, therefore I propose the following strategy for managing these file

Many times you can tell when your system is getting ready to fail. I contend that if you review your log files (with LOGERR) weekly and establish a known frequency of errors that exist you can use this base knowledge to determine if a device failure is waiting to occur (if you normally have three disc errors logged a week and all of a sudden you see ten logged, this should alert you of a possible abnormallity). Normally the error count will continue to increase until either a DISC I/O ERROR or system failure results. Some disc errors are also normal (like after a power fail).

For the tape drive, if errors increase you can take proper action before a failure results. Tape errors can be caused by improper cleaning, old tapes, or possibly a physical hardware problem with the tape drive.

Much time can be saved by effectively using the stream facility. It also saves us from having to explain the dialog which takes place. A utility in the TELESUP ACCOUNT will help in this area (IOERR).

V. SYSTEM FAILURE AND RECOVERY

I am often asked by customers when they can expect their next preventative maintenance (PM) visit on their hardware. I respond by asking when their last PM took place and emphasize that it is to their benefit to ensure the PM takes place on a periodic basis. Other components which can make a difference include: proper power conditioning equipment to ensure good clean power is fed to your system, a pleasant operating temperature (typically slightly cooler than you'd like), humidity control (especially helpful in the printer area) and a clean computer room (dust is bad).

Understanding the options which are available when a system backup takes place can also be of help. I would always perform SYSDUMPS specifying "DUMP FILE SUBSETS". This option tells the system the order in which to store file sets. Place the most important file sets before specifying all other files (@.@.@). After a failure, if a reload is necessary, you would perform a RELOAD (ACCOUNTS) followed by a RESTORE of @.@.@. This would allow you to begin work on those important file sets, which are a subset of all the files on the system, immediately after they are restored.

Remember the section on LOG FILE MAINTENANCE? The listing produced by LOGERR is very helpful in prediction of a hard failure (system failure). Another helpful program is MEMLOGAN. This utility reads the MEMLOG file and gives you a list of memory errors which have been corrected. To run MEMLOGAN you must be logged onto PUB.SYS. For more information on MEMLOGAN see the MPE SYSTEM UTILITIES MANUAL.

What happens if your not backed up and a failure occurs? Rarely does a system failure occur right after a normal full system backup. This being the case we need an approach to recover work in progress. SADUTIL is a handy supported utility which allows you to read files from disc and write them to tape when MPE is not functioning? SADUTIL is documented in the MPE UTILITIES MANUAL and runs stand alone (doesn't use MPE). This utility should be used before you need to depend on it so that procedures are well in place. How about dumping memory? On Series II/III systems there is microcode which copies memory to tape (memory dump) when certain buttons are pushed. On the HP-IB systems a file called SDFCOM lives in the PUB group of the SYS account which contains commands that determine what happens when DUMP is typed or the DUMP BUTTON depressed. This file can be accessed using the EDITOR. Once modified and kept as a permanent file this file can be used within the SYSDUMP dialog to update the version of "SDFCOM" which is on the SYSDUMP tape. A COLD START from this SYSDUMP tape will incorporate the new SDFCOM file on the system disc. The procedure to modify the SDFCOM file is listed in the CONSOLE OPERATOR'S GUIDE (Appendix O). The available commands interpreted by the SDFCOM (software dump facility) are listed in the same Appendix (O2). When viewing your particular SDFCOM file in the EDITOR make sure you understand what it will cause the system to do. When the first command in the SDFCOM file is a HALT your system will HALT when DUMP is typed or the DUMP button is depressed. Once this occurrs you must either depress the RUN button or type "RUN" at the console (followed by a carraige return) to allow the software dump facility to continue.

I recommend that the "HALT" be taken out of the SDFCOM file. With this done whenever DUMP is typed or the DUMP

BUTTON depressed a DUMP will occur (assuming LDEV 7 has an additional class name of DDUMP). I set up all my HP- IB systems with two commands in the SDFCOM file: DUMP and WARMSTART. Remember to use PROPAGATION for changes to the SDFCOM file. That is: COLDSTART from the PROPAGATED OFFICIAL COLD LOAD TAPE then make the SDFCOM file changes using the EDITOR, perform a SYSDUMP to place the new SDFCOM file in the MPE portion of the SYSDUMP TAPE, and finally COLDSTART from this SYSDUMP tape. This newly created SYSDUMP tape should be labeled as the latest PROPAGATED OFFICIAL COLD LOAD TAPE.

FAILURE RECOVERY

After a system failure, halt or hang an accurate list of environmental and visual indicators should be made. In the environmental area look for: power outages, temperature fluctuations, and static. Visual indications would include questions like: Are all the discs on-line and is the system in a run or halt state. Also copy any messages which were printed at the console prior to the system interruption.

Take a memory dump by either depressing the proper buttons on the SERIES II/III or by typing DUMP or depressing the DUMP BUTTON on HP-IB systems. The tape created should be dated and labeled with the environmental and visual indications.

At this point you're ready to begin the recovery process. A WARMSTART should be used to recover any spoolfiles. Use the "SPOOK" utility (UTILITIES MANUAL) to copy any spoolfiles to tape. Next, SHUTDOWN the system and COLD START from the PROPAGATED OFFICIAL COLD LOAD TAPE to ensure MPE is stable after the failure. After the COLD START has completed we can run SPOOK again to retrieve the spoolfiles previously stored to tape.

If the WARMSTART fails, this implies that something within MPE was destroyed by the failure. Your next step is to try to bring the system up from the PROPAGATED OFFICIAL COLD LOAD TAPE. Suppose the COLDSTART also fails? Before even considering a RELOAD try the COLDSTART again. If this still fails try the COLDSTART from the most recent SYSDUMP tape set.

If all the above gives no results a RELOAD is necessary. Before doing anything you might consider calling PICS for consultation. At this point you will lose all the work accomplished since the last backup unless you use the SADUTIL program. This stand-alone utility will save files to tape when MPE is inoperable. Using the first reel from the most recent backup perform the RELOAD specifying the ACCOUNTS OPTION. This will cause the entire accounting structure to be rebuilt with no user files restored.

If SADUTIL was used you must first restore the RECOVER2 utility (described in the MPE UTILITIES MANUAL) to read the files stored to tape by SADUTIL. This utility is on the ORIGINAL OFFICIAL COLD LOAD AND SUBSYSTEMS TAPE created for you when you were updated to your current version of MPE. Finally, you would use the RESTORE command to RESTORE *T;@.@.@;KEEP;OLDATE. This restore should start from the most recent SYSDUMP tape set and end with the last full SYSTEM BACKUP. If there are greater than 4000 files on your backup tape set, you must also use the FILES= parameter.

A final thought on failure recovery has to do with temporary file space. After a system interruption occurs a RECOVER LOST DISC SPACE should be scheduled. The time involved to perform this function depends on the number of files on your system. This recovery takes 5-10 minutes for every 1000 files. It is best to perform the recovery at the end of the day typically after the system backup. After the backup you would SHUTDOWN the system and COOL START with RECOVER LOST DISC SPACE. Recovering lost space will free up temporary space ($OLDPASS FILES or SCRATCH FILES) which was not released prior to the system failure. Performing this recovery will rebuild the FREE SPACE BIT MAPS on all disc drives from the information contained in the MPE FILE LABEL. If a FILE LABEL is not as it should be, the RECOVERY will alert you of the files in question and purge them from the system. Using RECOVER LOST DISC SPACE after a failure should give you an additional feeling of confidence that your system files and directory are safe.

VI. PERFORMANCE CONSIDERATIONS

To achieve the performance your system is capable of requires a dedicated effort. You must pay attention to specific resources that your applications require and organize a plan to maintain these resources. You can be guarenteed that not managing your resources can eventually result in poor system performance and/or system problems.

DISC FREE SPACE

The FREE2 utility should be used on a weekly basis to manage free space and monitor fragmentation. When discs are greater than 80% full some form of management should take place. If you are in a multi-disc environment each disc should carry a portion of the system file load. The best way to ensure a balance is to keep sufficient space available on each disc. Day to day activity will fragment the available free space. I suggest the use of VINIT to manage fragmentation on a weekly basis. After performing a full system backup VINIT can be used to CONDENSE each disc (VINIT is documented in the MPE SYSTEM UTILITIES MANUAL). Once a month a full system reload should be performed. Use the ACCOUNTS OPTION when reloading followed by a RESTORE of all files. Always verify the tape set you plan to reload from (see the section on STORE/SYSDUMP TAPE VALIDATION).

FILE PLACEMENT

Distribution of files can make the difference between good and great performance. A balanced access for files is what we would like to achieve with file placement. Placing files that are accessed together on different discs will balance the I/O load. A good example is the MASTER/DETAIL relationship in IMAGE DATA BASES. Placing the MASTER on one disc and the DETAIL on another will minimize head contention thus improving performance. Using the MPE STORE command, files can be written to tape, then using the MPE RESTORE command with the DEV= parameter files are brought back into the system to a specific disc drive. Specifying a specific LDEV # or UNIQUE CLASS NAME will cause file placement to occur (the supported program LISTDIR2 or the LISTF ,-1 command can be used to determine which disc a file is located on). Once file placement is achieved it is critical to perform RELOADS using the appropriate option. Using the COMPACT OPTION will place files on the same LDEV they were originally on. Using the SPREAD OPTION will place files on the same CLASS NAME they were originally on. If a RELOAD ACCOUNTS is choosen followed by a RESTORE @.@.@ those files restored to either a unique CLASS NAME or a particular LDEV # , will again be placed on those devices. All other files which have DISC as their file location will be SPREAD among the devices with CLASS NAME DISC (similar to the SPREAD OPTION).

Class names are also important for allocation of temporary space used for VIRTUAL MEMORY and SPOOLFILES. VIRTUAL MEMORY should be allocated on each disc equally. When allocation is set up in the SYSDUMP dialog you must specify how much VIRTUAL MEMORY will be assigned to each disc (this can be done by CLASS NAME or LDEV #). SPOOLFILE space is assigned to discs with class name SPOOL. Similar to RESTORE (which restores files by default to class name DISC), SPOOLFILE space will be spread to discs in the same ratio which the CLASS NAME is found for each disc. So if you give LDEV #2 the CLASS NAME SPOOL twice two spoolfiles would go to LDEV #2 before another disc would be searched for CLASS NAME SPOOL. The same cyclic pointer approach is used when files are restored with the RESTORE COMMAND (i.e. if a disc has a class name of DISC a file will be restored to it, if CLASS NAME DISC appears multiple times multiple files will be restored to it before another device is searched for CLASS NAME DISC.

Since files can be assigned by using the DISC CLASS NAMES DISC and SPOOL a scheme can be developed to reduce the number of accesses to the SYSTEM DISC (LDEV #1). In a multiple-disc system all discs except the LDEV #1 should be configured with CLASS NAME SPOOL AND DISC. This keeps spoolfiles and files being restored from going to the system disc. Place files which are rarely used on LDEV #1 so it is free from unnecessary accesses.

SYSTEM TABLES

Configuring MPE TABLES unnecessarily high will waste available memory. Using the guidelines in APPENDIX C of the SYSTEM MANAGER/SUPERVISOR MANUAL to configure TABLE SIZES is a good starting place. The unsupported utility TUNER4 can also be used to get a dynamic view of how SYSTEM TABLES are being utilized. If there is program development going on I also recommend that the CODE SEGMENT LIMIT be set to 8K. This will encourage programmers to segment their code, resulting in better memory utilization. DATA BASES

Using a DATA BASE on the HP3000 makes good sense. Once the DATA BASE has been developed careful attention should be given to the placement of the file sets which make up the DATA BASE. To reduce head contention MASTER AND DETAIL SETS should be located on different disc drives. Periodic review is also necessary to assure data set capacities are adequate. Prime numbers should be used for MASTER SET CAPACITIES and data sets should be less than 80% full. DATA BASE maintenance should also be performed periodically. The utilities DBUNLOAD and DBLOAD allow restructuring of the DATA BASE so that a performance improvement can be realized. When a DBUNLOAD CHAINED is performed entries on the primary path are written to tape in a contiguous fashion. Using DBLOAD will now load the data into each set as it was written to tape. Future accesses to your data by the primary path will now occur faster than they did before since all data entries are within contiguous blocks. If your site has the RAPID products the DICTDBU (unload) and DICTDBL (load) accomplish the same functions as DBUNLOAD and DBLOAD with greater sophistication. There is also a DICTDBA utility which provides AUDIT capability.

VII. HELPFUL UTILITIES

There are a large number of UTILITIES available which can be very helpful in managing your system. The HP INTERNATIONAL USERS GROUP produces a set of UTILITIES (USERS CONTRIBUTED LIBRARY) available to its members. I've decided on a select group of UTILITIES which I feel should be on every HP3000 system. The majority of the UTILITIES I've included were written by HP personnel to solve problems which the current set of SUPPORTED UTILITIES does not address. HP SUPPORT DIVISIONS are in the process of reviewing those UTILITIES which they feel could benefit the expanding customer base they will be in an account called TELESUP.

HPUNSUP GROUP IN THE SUPPORT ACCOUNT

The UTILITIES I will include on your system are UNSUPPORTED. The HPUNSUP GROUP should be given special capabilities to allow running of the various programs. Place a password on the SUPPORT ACCOUNT and review the FILE ACCESS SECURITY for each of the files so you understand who has the ability to READ and EXECUTE them. Create the GROUP like this: NEWGROUP HPUNSUP;CAP=IA,BA,PH,PM,MR. Some of the programs in this GROUP should not be executed until you've discussed the problem with your ACCOUNT SE or PICS.

UTILITY DOCUMENTATION

BADLABEL....(run only when asked to do so)....BADLABEL, checks each file label on the system and produces a tape which is sorted and checked for any file extents which point to other file extents. It then produces a listing, to $STDLIST, of any bad file labels. Using BADLABEL with the NOTAPE entry point will cause the cross reference of file extents to be bypassed. To run BADLABEL without a tape you would enter: RUN BADLABEL,NOTAPE.

BULDACCT....Program to recreate the ACCOUNTING STRUCTURE of your system. It is useful for keeping your ACCOUNTING STRUCTURE in a job stream format.

BULDACTD....Documentation on running BULDACCT. DBLOADNG....A program which displays information on DATA BASE loading. Read DBLOADOC for details.

DBLOADOC....Documentation on the running and interpretation of DBLOADNG.

FLUTIL3....(run only when asked to do so)....File label patch utility allows the user to modify the contents of a file label. DIRECPURGE will allow bad file labels to be deleted from the system directory. File space taken up by the file deleted is not returned to the system. A RECOVER LOST DISC SPACE is needed to retrieve the lost space.

LOGAUDIT....A program to view the contents of a tape created by LOGSNAP. This program will also view LOG FILES on disc. Events viewed are the same events LISTLOG2 views (LISTLOG2 is documented in the SYSTEM UTILITIES MANUAL).

LOGERR....A program which interrogates LOG FILES for I/O ERRORS.

LOGMESS....This message file is needed for LOGERR to function. It should be moved to PUB.SYS for LOGERR to function.

LOGSNAP....A program which copies LOG FILES to tape.

PSCREEN....A program to copy a terminal's memory to the line printer. It's useful in documenting error messages which are printed on a terminal.

RESTORE....A program which allows you to restore files from a previously created STORE TAPE into your current log on group and account.

SYSINFO....A program which allows you to view the system I/O configuration and tables without using SYSDUMP. It allows easy printing of configuration information to the printer.

TAPETEST....A program which writes all 1's to tape and reports when it cannot read the 1's it previously wrote. This program provides a rough check for tape quality.

TUNER4....A program which gives a dynamic view of system table usage.

VALIDATE....A program which reads a tape created by STORE or SYSDUMP and reports the directory and file labels read.

JOHN VEGA--HP SYSTEM ENGINEER 9606 AERO DRIVE, SAN DIEGO CA. 92123

(619) 279-3200 EXT. 193

STARTED WITH HP: SEPT, 1975

POSITIONS HELD: CUSTOMER ENGINEER FOR 2 YEARS

TECNICAL SUPPORT ENGINEER FOR 3 YEARS

SYSTEMS ENGINEER FOR 3 YEARS

SPECIALTY: DATA COMMUNICATIONS BOTH HARDWARE AND SOFTWARE

DISTRIBUTED PROCESSING USING IMF/3000

by Michael E. Ura

Over the past five years, much has been written about distributed processing. The concepts are exciting. The ability for a user to sit down at a terminal to access and process data no matter where it resides is certainly the key to success in building a distributed system. Unfortunately, translating distributed concepts into real-world, cost-effective application systems is very difficult. Moreover, vendors have only been able to supply the "tools" for building distributed systems. Very little software is on the market to support true distributed system technology. This paper will address the use of several HP tools, most significantly IMF/3000 (Interactive Mainframe Facility) to create a distributed Inventory Management System utilizing large- scale IBM mainframes (IBM 30xx) and HP/3000 series computers.

To bring the paper into perspective, a brief description of the company's environment is necessary. Monsanto is a large ($6 billion plus in sales) chemical manufacturer consisting of five operating companies each with plants located throughout the United States. Each operating company is responsible for formulating, within reason and corporate guidelines, a hardware and system strategy. The Monsanto Industrial Chemicals Company (MIC) has chosen the HP/3000 Series of computers and its associated software to support Information System Processing at headquarters and each plant location. Figure 1 is a layout of the present network. These HP computers provide each plant location the ability to develop and process in-plant systems independent of headquarters while continuing to feed and access companywide systems when necessary. Implementation of the HP/3000 strategy within MIC began in 1981 and will be completed in 1984.

As with most companies, inventory control is a key function. The responsibility for inventory control in Monsanto is delegated to each operating company with the only corporate requirements being the reporting of monthly inventory balances for inclusion in the General Ledger System. Figure 2 outlines MIC company's inventory control function. As can be seen, inventory information and reporting requirements exist throughout the organization, both in a central location and at each plant location. Because of this central need, and the state of computer technology in 1977, the decision was made to design a large inventory data base system on the IBM mainframe, utilizing IBM's product IMS (Information Management System). The system was essentially batch oriented with no on-line support facilities, even though the IMS software was fully capable of handling on-line communications. Plants were required to use the system to report and manage their inventories. While the system served the headquarters inventory needs very well, it was incapable in its batch mode to keep up with the daily inventory needs of the plants -- the most important inventory control point. As MIC entered the economic recession of 1981 the need for better inventory controls across Monsanto became evident and the MIC plants cried out for help. The question at that point was, "Do we scrap our present Inventory Control System, a $1.3M investment, and utilize our now emerging HP/3000 strategy?" or "Do we attempt to build on our Inventory Control System and fit the HP/3000 in somehow?"

This was not an easy decision to make. Some pertinent issues concerning the predicament will make the transition to the use of the HP/3000 in a distributed mode more evident.

1. The present Inventory Control System was basically good. Its data base was well designed and its programs were virtually error-free. All company and corporate interfaces already existed and worked well.

2. The inventory information needs of the users at headquarters were being met.

3. The inventory information needs of the plants were not being met. The batch mode of processing was too slow and costly to meet the demands of the plants.

4. Each plant operated differently. Their inventory information needs as well as the way they processed inventory information was different. In some cases the only common element was that they received raw materials, produced products and shipped the products to customers.

5. Having an HP/3000 strategy approved, it was not feasible to recommend any other vendor.

Issue four was the key to the decision to build a distributed system. A major requirement put forth by the plants was that any system must meet the needs of the plant; therefore, a common solution was not acceptable. But how to fit a centralized system into a plant-unique environment is not an easy question to answer, especially in an existing system so dominated by IBM technology. The key to the solution was HP's Interactive Mainframe Facility (IMF/3000).

HOW THE SYSTEM WORKS

IMF/3000 operates in two modes, pass-thru and programmatic. Pass-thru simply makes an HP26xx terminal look like an IBM 3270- type terminal. A communications line from the IBM mainframe is attached to an HP/3000 with IMF/3000 installed. IMF makes the HP/3000 look like an IBM 3274 control unit, giving all HP26xx terminals the ability to look like an IBM 3270 terminal (CRT) device. This facility allows a user to not only access the HP/3000 but also allows him to access facilities on the IBM mainframe such as TSO, CMS and IMS application systems. The end result is that a user does not have to have two CRT terminals on his desk if he requires access to both the HP and the IBM. Figure 3 compares a pass-thru environment with an IBM 3270 environment.

IMF/3000 programmatic access emulates an IBM 3270 environment much the same way except that a program is written, usually COBOL, to control inputs and outputs of the user in conjunction with inputs and outputs of the IBM mainframe. With the ability to control both the HP and the IBM communications the problem of fitting plant-unique processing into a centralized system was neatly solved.

The end result was the creation of a multi-plant inventory system with a centralized data base residing on an IBM mainframe with distributed processing of that data by multiple site HP/3000's in an on-line mode. Access to the data base is provided using the IMF/3000 programmatic links to IBM's program product IMS (Information Management System) with screen formatting provided on the HP via VIEW/3000 and resident HP files using KSAM/3000. Figure 4 provides a pictorial view of the system and its files.

As can be seen in figure 4 there are essentially four parts to the system: IMS, IMF/3000, VIEW/3000 and the preprocessor. Each of these will be discussed in detail then will be put together for a general discussion of how the parts interact as a whole.

IBM'S INFORMATION MANAGEMENT SYSTEM (IMS)

IMS, as typical of most IBM products, is very complex. It provides for the creation and maintenance of data bases; batch and on-line access to those data bases and provides a language (DLI - Data Language I) for accessing data bases from COBOL, PLI and Assembler language programs. Two concepts are key to understanding the use of IMS in the distributed system. First, the Inventory Data Bases are IMS data bases and therefore must be accessed by programs on the mainframe normally referred to as IMS programs. In the Inventory Control System these programs are COBOL programs that access the data bases using subroutines (or technically DLI calls). These subroutine calls are very similar to IMAGE data base intrinsics on the HP/3000. The second concept is that of on-line program execution. Programs that are to execute in an on-line mode reside on special program libraries on the IBM defined to IMS. Each program is assigned an eight-character transaction code. This transaction code identifies not only the program to be executed but also identifies which data bases are to be accessed. When IMS receives an input message over the communications line -- a message looks like this:

-- the transaction code is interrogated and the proper program is executed. The program then simply acts on the data received and returns a new message to IMS which in turn sends the outputted message back to the same source as the inputted message. This source is normally a user sitting at a terminal but in our case is the IMF/3000 software on the HP/3000. Under the distributed system the appropriate message is simply sent to IMS and received from IMS via the IMF/3000 software. Program execution including queue time is normally less than three seconds with overall response time in the 5-7 second range dependent on communication line speed and volume of data being passed on the line.

HP'S INTERACTIVE MAINFRAME FACILITY (IMF/3000)

IMF/3000, as mentioned earlier, provides a means for controlling communications (data to be sent across the line) between an HP application program and an IBM mainframe. An application program using IMF looks to the IBM as if it is a 3270 CRT terminal; therefore, the application program must tell the IBM what it is doing by calling IMF intrinsics. The majority of the intrinsics are fairly simple to use. The following is a list of the most commonly used intrinsics:

OPEN3270	Open communication between the application program and the communication line
STREAM3270	Puts data from the application program into the IMF buffer
TRAN3270	Transmits the contents of the IMF buffer to the IBM host
RECV3270	Receives data from the IBM host and puts it in the IMF buffer
READSCREEN	Gets data from the IMF buffer for the application program to use
CLOSE3270	Terminates communications between the application program and the communication line

There is also an intrinsic, ACQUIRE3270, that actually allows you to start an IMF pass-thru session from your program with the program regaining control upon exiting of pass-thru mode.

While these intrinsics look simple and are easy to code, talking to an IBM host via IMF is not easy. The key is knowing exactly how the IBM will respond in any given situation. For example, the application program may send an IMS message to the IBM upon execution of a TRAN3270 intrinsic. The program will then immediately execute a RECV3270 which waits for a response from the IBM host. When a response is received the program executes a READSCREEN and, Guess what? You didn't get what you expected. The reason for this is that sometimes, but not always, the IBM will simply send an acknowledgement that it received the data sent. In this case the first message is discarded with a second RECV3270 and READSCREEN returning the expected response. In other cases a system broadcast message such as "THE COMPUTER IS GOING DOWN IN 5 MINUTES" may be received. Obviously the user needs to see this message; therefore, it cannot be discarded. Again, a second series of RECV3270 and READSCREENS will result in the proper data being received from the IBM host.

This is just one example of the problem in using IMF intrinsics. For those familiar with IBM on-line systems the intrinsics work excellently with IMS but very poorly with TSO and CMS. The reason for this is that in IMS the HP application program can control the IBM. Under TSO and CMS the IBM controls you. It is important when programming with IMF

intrinsics that the programmer know the use of these intrinsics inside and out (the HP IMF class would be helpful). Secondly, the programmer needs to realize that much trial and error will be required in tracking down when and how IBM will respond. But once the series of transmits and receives are worked out the program will work almost flawlessly. Upon completion of the Inventory Control System actual subroutines were developed to handle all HP-IBM/IMS communications. A single call to the subroutine TRANSIMS contains all the logic necessary to send and receive messages while handling all possible error conditions. The data to be sent to the IBM for processing is passed in the subroutine call with the subroutine returning the processed data. Future generations of programmers have been spared countless hours of trial and error testing.

V/3000 - VPLUS/3000 - VIEW/3000 OR WHATEVER HP NOW CALLS IT

V/3000 is used in the Inventory Control System for all screen processing. V/3000 intrinsics are called from the preprocessor program to read, write and edit screens. Since many papers have been presented to the IUG on V/3000 in the past further discussion as to the inner-workings of V/3000 is not really necessary.

The key to using V/3000 in the Inventory Control System is in the require- ment for plant-unique processing. Since each plant is different, many of the differences in processing are simply in the inputting of data. For example, Figure 5 represents two different plant "packing" transactions. (Packing represents the process of taking material produced from a silo and putting it into bags, drums, etc.) V/3000 as a screen processor works very well in this area. Each plant therefore has its own V/3000 forms file to hold its unique forms.

THE PREPROCESSOR

The preprocessor is a COBOL program that runs on the HP/3000. The reason it is called a preprocessor is because it accepts plant-unique data and formats it into common transaction (message) formats for processing by the IBM host. Each plant using the Inventory Control System has its own preprocessor just like it has its own V/3000 forms file. This issue of common transaction formats and preprocessing of data is what ties the system together as it relates to plant-unique processing. Taking our packing example from V/3000, Figure 6 shows the relationship between the plant's unique format and the standard transaction format. This use of standard transaction formats allows the plant, through V/3000 and its preprocesor, to create any type of input screens or put any data relationships, table lookups or special plant coding that is needed into the preprocessor. As long as the proper standard transaction is produced, the plant can do about anything it wants in the preprocessor, including the updating of plant-required data bases resident on the HP/3000.

THE PREPROCESSOR - PUTTING IT ALL TOGETHER

Basically this is how the preprocessor works and interrelates with IMF and V/3000:

1. The user selects a transaction, such as packing, from the master menu.
2. The packing screen is displayed on the CRT and the user inputs the necessary data.
3. The screen is then read using V/3000 intrinsics with V/3000 field edits taking place.
4. If all V/3000 edits are passed the transaction is then put into its standard transaction formats and the IMS transaction code is applied.
5. The transaction is then sent to the IBM mainframe using IMF intrinsics.
6. The IBM mainframe processes the transaction, either updates the data base or gets inquiry data depending on what the user is doing, while the preprocessor waits for a response from IBM that all processing is complete.
7. The transaction is then received by the preprocessor using IMF intrinsics.
8. The preprocessor unformats the standard transaction returned into the plant's unique format.

9. The screen is then redisplayed showing that the transaction was accepted or that it was rejected with the proper error messages displayed in the V/3000 window.

This logic applies to all transactions where one message is sent to the IBM and only one is received. In the case of inquiries one message is sent, such as a request for all open orders, and many messages are received. In this case, during step 7 above, the returned messages are written to a KSAM file as they are received from the IBM host and then unformatted as needed in step 8.

FUTURES

As can be seen from the above steps, true distributed processing is taking place. Data formatting and editing is taking place on the HP/3000 with actual data base interaction taking place on the IBM. IMF/3000 has taken what used to be done in batch mode and allowed it to be done in an on-line mode.

But what is equally important as the use of IMF in distributing Monsanto's Inventory Control System and the potential to use the same techniques in building future systems requiring HP-IBM interaction, is the use of the same concepts in a strictly HP environment. For example, a data base system using IMAGE could be built on one HP/3000 with other HP/3000's accessing and process- ing data from that data base much in the same way as the Inventory Control System does. In MIC these concepts are being studied for use in interconnect- ing HP/150 personal computers to HP/3000's. Additionally, IMF is being used to transmit documents generated from HPWORD to IBM/5520 word processors via the IBM mainframe and CMS.

Concepts are the key, however. The use of the IUG forum to present specific applications and concepts for future systems growth is invaluable.

Michael E. Ura is a Principal Systems Analyst in Monsanto's Industrial Chemical Company located in St. Louis, Missouri. His nine years of MIS background include operations, programmming, systems analysis, project management and consultation. Mr. Ura has a B.S. degree in Business Administration from the University of Missouri and a Masters degree in Management from Webster University.

LOCAL AREA NETWORKING SIMPLIFIED WITH A DATA PABX

BY C. H. "Bill" Skaug

Micom Systems is a rapidly growing manufacturer of data communications equipment for minicomputer users, and, as is typical of growing firms, our data processing systems have been expanding even faster than the company itself. In fact, in terms of the computers we use, the remote sites they support, and the communications links between them, we quickly became a very typical prospect for local networking.

Four years ago, when our company was less than a fifth of its present size, we met our processing requirements by using service bureaus. Today we have six in-house computer systems and three others at remote sites -- in Pennsylvania, Puerto Rico, and England. All are interlinked.

We fortuitously made some sound decisions in our very early stages of growth, and have since managed to avoid much of the pain that frequently accompanies major changes in installed networks. As a result, we believe we can set up some trail markers for others who might be progressing along the path we took: from dial-up lines, to data concentrators and dedicated in-house links, to data PABX-centered local networking, and finally to multiple interlinked nets.

Beginning with dial-up access

Starting in 1980, we used two major computing services, Xerox Computer Services and General Electric Information Services Company (GEISCO). Xerox supplied standard "canned" business applications, while GEISCO was used on a more ad hoc basis for engineering and management applications. Due to the different usage patterns of the two services, different communications techniques proved appropriate.

For the quick or intermittent problem-solving performed by a large user base accessing GEISCO, we used dial-up lines, allowing the engineer or manager to reach the system from a terminal in his (or her) office. Dial-up access is one of the simplest forms of data communications, requiring only a modem to supplement the user's terminal, and this is the way many organizations begin their dp. We were no different.

Concentrating on Leased Lines

We used the Xerox Service Bureau for the day-to-day commercial applications, which typically were accessed by a smaller group in the accounting and manufacturing departments. These users were on-line all day, or at least a great part of it. This meant that a leased, rather than dial-up, line would be most cost-effective.

To further keep telephone line costs under control, we installed 8-channel (later 16-channel) data concentrators at each end of the leased line. These concentrators (sometimes called statistical multiplexors) let us run up to 16 terminals while paying for only one telephone line.

In some ways, using concentrators and a leased line is at least as simple as dial-up access, since the amount of data communications hardware is reduced. Instead of 16 modems at our site and another 16 at the service bureau, plus 16 phone lines, our communications link had one concentrator with an integral modem at each end of the single telephone line. And the concentrators could pay for themselves in a matter of months in savings on telephone line charges).

Driving an In-House System

Then, about four years ago, when we were a $15 million company, our growing size and requirements finally sent us looking for our first in-house computer. Perhaps naively, we hoped for a single system that could handle all our dp needs: administrative, financial, engineering software support, customer service, and CAD/CAM. We installed a Prime 750, mainly due to software considerations.

However, by the time the Prime was up and running, we had already outgrown it. We proceeded to order our first HP 3000, a Model

44 (again chosen due to its software), along with another, smaller, Prime to handle CAD/CAM on a dedicated basis. We had decided to put financial, customer service, and manufacturing applications, including MRP and Bills of Material, on the HP, leaving the Primes for use by our engineering groups.

Fortunately for us, with the installation of our first computer, the Prime, we made a data communications policy decision which is with us today: unless the terminal is in the same room as the computer, we use line drivers to connect the terminals to their hosts. For those who are not familiar with them, line drivers are functionally analogous to modems but much less expensive. Capable of operating over distances of a few miles, they condition the signals going between computers and terminals in order to ensure reliable transmission beyond the EIA RS232C requirement of 50 feet.

While people routinely exceed the 50-foot limit without modems or line drivers, we didn't want to risk the problems of cross-talk or electrical interference when our walls and ceilings got more crowded with active datalinks. In fact, we even use line drivers to connect terminals to nearby data concentrators if there is any chance of a problem.

Accessing Multiple Computers

When our second computer, the HP 3000, arrived, we faced a new problem, one which would compound as we added more machines. While we had successfully managed to split applications between HP and Prime systems, many users needed access to both. So much for "best laid plans."

Faced with providing one user access to two computers, we had several choices. We could have added extra wires through walls and ceilings for each multi-machine user, and asked the user to plug his terminal to the appropriate set as needed, or we could have--even more magnanimously--installed a second terminal for each of these folks. Neither solution was economically viable, particularly when we envisioned the consequences of adding more people and more machines to cover our continuing growth.

Instead, we left the wiring and terminal situation as it was, and made our move in the computer room, where we installed a Micro 600 Data PABX. With the data switch, as it is often called, we have in essence a private telephone system for data that can connect any authorized user to any available computer port. And, if a port isn't available--say all the HP ports are tied up with accounting business at the end of the quarter--then the user asking for an HP port is told (in effect) "They're all busy. Wanna wait? You'll be number three in line." (This is the data PABX equivalent of camp-on busy in a voice exchange.)

The Advantages of a data PABX

The primary advantage of a data PABX, although it has many, is its ability to connect any terminal to any requested resource (subject to security considerations programmed into the switch). This means we can hardwire a terminal to the data switch, and from the terminal keyboard a user can request a port on one of our HP systems, one of the Primes, a Zilog development system our engineers use, or whatever we may acquire in the future. We can add more computer systems and terminals as we grow. We can even let users call outside, through the switch, to reach one of the service bureaus.

We also benefit from a "statistics log" feature of the switch which provides detailed reports of all switch activity (a function which would require a dedicated processor in other proposed networking plans). And our network easily supports remote users and remote computers.

The switch also lets us save money by using leased lines, while still providing "dial-up like access" to several computers. And nothing keeps us from "remoting" CPU's, heightening that dial-up parallel.

An example may help to illustrate the degree of flexibility we realize by combining data concentrators, leased lines, and the switch. Say one user at Micom Caribe, our Puerto Rican facility, needs to use one of the HP systems. He simply turns his terminal on, and the switch, which is in Chatsworth, California, the central site, automatically asks him where he wants to be connected. He can answer with a symbolic name -- in our case with an "H" for one of the HPs or with an "M" for the HP running MANMAN, or whatever -- and the data PABX makes the connection.

A few minutes later, an engineer in Puerto Rico may turn on his terminal and ask for one of the Primes. Although both active terminals are on the same leased line, each user gets the same service he would on his own unshared line. For that matter, either of the two users can log off one machine and ask for another without affecting the transmission of the other user on the line.

Growing the Network

As might be expected, as our data processing capabilities have grown, so has the data PABX. Luckily, the switch is easy to expand with the addition of simple plug-in interface card modules. These provide four line or port interfaces per card slot, and each bay has up to 32 slots, which works out to 128 lines or ports (intermixed) per 19-inch bay.

As we fill a bay, we simply add another. Once we had our first HP 3000 and Prime pair connected to the switch, it took about 30 days before we added our second bay; 60 days later we put in bay number three. We've already installed our fourth, and we can continue to put them in until the raised floor collapses from the weight.

All of this can lead to a massive set of cables for local and remote terminals, we found. We have about 250 terminals in our Chatsworth facilities, and another 40 to 50 in the field. To simplify the wiring of the nearly 300 RS232C connections coming into the computer room, we have adopted a technique called "group termination." Incoming lines go to a wall-mounted telephone block; cables with 50-pin connectors attach the block to the switch. Each of the cables handles either six terminals (with EIA control signals) or 12 terminals (data only).

Of course this greatly reduces the snarled wiring behind our switch, and decreases the cabling under our flooring. It also makes connecting to the data PABX quicker, as we can attach as many as 12 terminals when we plug in a single 50-pin connector.

Growing New Networks

Another advantage of basing local networks on data PABXs comes from the transparency of the network and the flexibility for adding new network components. While our switch grew, we acquired a pair of used HP 3000 Series III systems. One has been traded up to a Model 64. The other is used for program development in Chatsworth.

The latter HP3000 was used to develop applications required at Micom Caribe. While our MIS department developed the programs, users in Puerto Rico could monitor our progress and offer feedback, as they, too, had access to the system through the data PABX. This arrangement allowed us to maintain centralized control over the development process, and we also avoided the problems of flying liason staff members back and forth between the two sites.

We originally thought this HP 3000 would be moved to Puerto Rico, and we planned to use the data PABX to smooth the cut-over: We would have put the Micom Caribe applications on one of our other HP 3000s, and have the switch automatically connect Caribe users to the new (temporary) host. We'd then ship the computer to Caribe, and fly the latest SYSDUMP of the Caribe applications to Puerto Rico once the hardware was installed and checked out. The actual cutover would take place on any convenient weekend.

However, since we found that we needed the computer for our own use, we wound up ordering another HP 3000 for delivery to Puerto Rico. Then, it was just a matter of sending the SYSDUMP tape and loading it over the weekend.

Even though Caribe now has its own computer system, we've retained our leased satellite link. We now use it to tie computer ports as well as terminals into our switch in California. This give users at each site access to each other's computer systems and data bases. And, when Caribe's workload warrants it, we can also install a data PABX there to attain still another degree of flexibility. An "interconnect facility" available for the data PABXs makes it possible to link switches, yet the resulting "network of networks" appears to the user as a single large data PABX.

And, there is more. We've brought our British subsidiary, Micom-Borer on-line with the installation of an HP 3000 model 40, our Black Box Catalog subsidiary in Pittsburgh, is coming on-line with a mini, and two other subsidiaries also link to Chatsworth for terminal access.

In Chatsworth, we plan to take the "interconnect facility" a step further: we're installing a data PABX in our customer service facility that will support local terminals there and also link To our data PABX in the headquarters building -- there will be no computer ports connected directly to this switch. Instead, the customer service data PABX will communicate with our central switch using data concentrators and leased telephone lines. Since communications between data PABXs is transparent, users still will have access to any needed resource just as if they were directly connected to the central switch, but this arrangement will allow us to use fewer telephone lines and data concentrators (compared to giving each terminal a channel on a concentrator). It also will mean that if a phone line fails, customer service's switch can automatically direct users' connect requests to a good line. In addition, we'll benefit from the statistics log of the customer service switch, which will readily tell us if we need more lines to accomodate demand, or if we have installed more lines than e really need.

Networks Within Networks

We're also expanding the scope of our networking in another way. With our third Chatsworth-based HP 3000 we began using HP's networking program, DS/3000. The new machine connects to our data PABX and (through a high speed link) to one of the other 3000s. Our reason for going to DS/3000 was one of response time for our many users in finance, manufacturing, and sales support who make inquiries against a large IMAGE data base originally residing on one of the 3000s.

We put MANMAN and Accounts Payable and General Ledger on one of the linked 3000s, Customer Service, Accounts Receivable, and Order Management applications on another. Files unique to each host became more readily available to our users, and common files are kept synchronized through the facilities of HP's networking program DS/ 3000.

Because of the transparency of the data PABX, we don't have any problems with the two networking systems (ours and DS/3000) operating together. Additionally, if someone needs to switch applications, say from A/R to A/P, it is more efficient to reconnect to the appropriate host through the switch, as opposed to using the high speed interprocessor link and stealing capacity from DS/3000. We also have DS/3000 capability on the Caribe HP 3000, so we can make bulk file transfers and synchronize data bases between California and Puerto Rico as well.

Conclusions

As we've grown our network, we've learned a few things, the most important of which is just how right we were to start with a data PABX. Unless your users stick to a single computer, or don't mind marching off to a centralized terminal room, it's difficult to see how to manage without one.

Perhaps it's inevitable that our version of local networking using a data PABX is continually compared to others which use more exotic technologies. We don't mind. We show up well in the comparisons. For example, the new proposed local networking techniques which use coaxial cable or fiber optics offer one very appealing feature in their ability to support many users on the same physical medium. But we can do the same by adding a few new twists to plain old telephone technology. For instance, in our corporate headquarters we've begun using a new type of line driver that can multiplex up to eight asynchronous terminals over the same two pairs of wire that might otherwise be used for a single telephone or terminal. This saves us time, money, and effort. We already have our offices wired for terminals, so now we can use the same wiring to connect several terminals in one room to the data PABX. In addition to saving us the time, cost, and disruption of stringing new wires, it also reduces the number of line drivers we need by as much as a factor of eight.

Should we wish to do so, a data/voice multiplexor is also available for use with the Micro600. (Called "Instalink", it allows a terminal and a telephone in the same office to share the existing single telephone line there without affecting each other's use.)

Granted, telephone technology is not as exotic or glamorous as working with coax or glass fibers, but it more than makes up for its lack of sexiness. Its technologies are proven, and relatively standardized in a de facto sense. Twisted-pair wiring or in-house telephone wiring is inexpensive, and in most offices, it's already in place. Likewise, using line drivers with RS232C interfaces provides a standard method of connecting to the network and avoids any special programming considerations.

In contrast, networks using coaxial cable or fiber optics run up increased costs due to the expense of the medium. In most buildings installing the broadcast medium also runs costs up very quickly, as well as disrupting everyone's work. Then too, the interfaces to the media are more expansive -- by an order of magnitude -- than line drivers, and are basically unstandarized, incompatible devices today.

A pioneering user who adopts one of these new and exotic networking technologies may well be casting his decision in concrete before the industry is ready for that. A mistake on his part is likely to become an expensive embarassment down the road. We're not going to have that problem. When those new technologies mature and users can install them with confidence, we won't be left behind; we can connect to them too.

Charles H. Skaug is Director of Management Information Systems for MICOM Systems, Inc., in Chatswrorth, California. He recently established a local network there, linking MICOM's several facilities in Chatsworth with the firm's 10 regional sales offices and its outlying subsidiaries in Pittsburgh, Puerto Rico and Great Britain. Before joining MICOM, Mr. Skaug had been U.S. Air Force Computer Installation Superintendent in charge of worldwide Air Force System Design projects, and had also served as Manager of Data Processing at TRW (Aertech).

Office Automation, A Cautious Approach

by Russ Bradford
Bradford Business Systems, Inc.

Office automation has been touted for the past several years and has now been thrust into the limelight through an intensive industry campaign to educate the business community. Unfortunately, through all of this hoopla comes no clear definition of what office automation is, or exactly what the benefits are supposed to be in terms of hard dollars and cents savings. This information will eventually surface after several companies have done some rather expensive experimenting and gone through the trial and error phase which all new endeavors must experience. This paper is intended not to dissuade the potential entry into office automation but, to the contrary, to encourage a more rational entry into this much needed boost to office efficiency and productivity.

Office automation IS needed but until all of the wrinkles are ironed out, purchases of specific hardware solely for office automation should be carefully considered to avoid the possibility of early obsolescence and incompatability with future developments. The major manufacturers of office automation systems are all coming up with their own methodologies, "standards", terms, networks and a variety of clever ideas. The single vendor solution was desirable in the past because one would only have to depend on that vendor for all support service. With the variety of approaches available, your vendor may have the wrong approach for your company, or one that just won't remain viable or cost justifiable in the years to come. I encourage you to shop around and become thoroughly confused by the cornucopia of devices, gismos, widgets and software packages that are supposed to make your company ten times as profitable as it is without them. No doubt, there are many inventions and programs that truly will save your company money, but care should be taken to get the right solution for today, yet with an eye toward the future as well.

Before we go on, maybe I should take a stab at defining the areas of a company that might be automated.

a. The secretarial pool or function has basically remained unchanged in any major way for over a century. A secretary had a typewriter and a steno pad in the past, and they have them now. It is true the typewriters of today are faster, easier, and quieter than their old counterparts, but they are basically the same machines. Copiers, dictation equipment and a few other pieces of office equipment have helped to streamline the secretarial function, but not too radically.

   Areas such as mass mailings, report generation and reformatting of long reports have been avoided in the past due to the large effort required to accomplish these tasks on conventional typing equipment.

b. Management of a company is the lifeline for success or failure of many companies. Often, a company's resources can usually only afford a given number of these more expensive individuals, and thus their time must be utilized as efficiently as possible. A good manager should be able to organize his day so that he can get everything done between 9:00 AM and 5:00 PM, but that isn't always possible since demand for his or her time far exceeds the supply in

most cases. Automation of some of the more time-consuming aspects of a manager's job can help to alleviate this bottleneck.

c. Communication is most likely the chief area of concern for managers and thus office automation should definitely address this aspect of the managerial role. Communication covers the areas of telephone communication, creation and distribution of memos, reports and meetings.

Ineffective communication can be the downfall of many well laid plans if the information needed is not received on time or is misunderstood by those who receive it. The areas of electronic mail, calendar scheduling, and teleconferencing are key ingredients of any office automation system.

Many managers are no longer alien to the thought of typing their own memos since they have found that they can type one in using two fingers as fast or faster than they could write it in longhand, and they don't have to wait for a secretary to type it. This reduces the time it takes to get a memo out. A typical scenario is for a manager to type something and then have a secretary turn it into English, since many managers can't spell to save their lives.

d. Meetings that require the attendance of individuals from diverse geographic areas are time consuming and expensive. Video conferencing can reduce or even eliminate the bulk of an executives time spent traveling for many meetings and conferences at far less expense.

e. Since the U.S. mail is expensive, slow, and unreliable the evolution of electronic mail is inevitable. Some bottlenecks still exist to prevent full implementation in the next year or two. We must first wait for the Washington political mechanism to make a few decisions that will affect the success or failure of electronic mail on a wide scale. It may be decided to let the U.S. postal service run an electronic mail system which could lead to numerous delays and undesireable changes in what might be a defacto standard developed by industry. For the time being most companies should be satisfied with a local form of electronic mail for use within the organization.

f. Scheduling of meetings and events as well as reminders becomes a problem that is easily solved through the use of computer software geared toward the calendar scheduling function. Some forms of this software can broadcast messages to users about upcoming events as well as allowing the users to leave themselves or each other reminders about meetings or things to be done.

g. Computer graphics has been around for a number of years but until recently has not ended up in the hands of the casual user. Business graphics can be an invaluable tool for reducing numeric information to an easily understandable form. Graphics might be considered a form of communication that is a valuable time saver in that it reduces the amount of time and effort required to convey relatively complicated information.

h. Training of staff is a major area of concern in many corporations due to the great expense and relatively high turnover of clerical personnel. Training programs are expensive and thus alternatives such a video training are being explored.

Basically the aforementioned fields are the major areas that we intend to address in this discussion. Office automation can reach into numerous other areas as well.

Word processing, electronic mail, teleconferencing, video training and calendar scheduling are the major areas that we feel will be the principal focus of the major manufacturers in the coming decade of rigorous development. To date the offerings in all of these fields have usually left the potential buyer apprehensive because there are gaps between the buyers' expectations and the vendors' solution in all but the most rare instances. An ideal word processor would be one that "intuitively" knew what the operator wanted, without the operator having to "tell" the machine every little detail. This would be nice, but at present and for a long time to come, if ever, word processors will need to be told what is required of them, thus creating a gap between some prospective

buyers' expectations of the ultimate and reality.

Electronic mail is another example of a good idea that is not quite ready for implementation. There is no widely accepted network in place and thus electronic mail is a tool that can only be used internally within a company for the time being. A company that spends a lot of money setting up a communication network with its related software and hardware, but doesn't conform to whatever twists MA Bell has in store is going to be left isolated. Despite all of the discussion about protocols, local networks, message switching and the like I am still apprehensive about where this is all going to lead in terms of the "true standard". The major manufacturers, fearing loss of ground by not taking a stand are trying to develop the best network for their own purposes with an eye toward what each believes will become the standard. Its a game with some very high stakes and both the buyers of such systems as well as the vendors are risking some financial setbacks if the assumptions about standards are way off base. An example of this syndrome lies in those companies who believed the myth that mainframes were the only way to go, and that minicomputer and distributed processing didn't stand a chance of being cost effective or viable solutions. Every HP customer must have some belief that distributed processing and minicomputers are here to stay and I totally agree. Companies that took the plunge and purchased that 2 million dollar mainframe with hopes of growing into it are now purchasing minicomputers to round out their systems and get computing into the field where it is most needed. I am not saying that mainframes don't have their place, but instead too narrow an approach to such a problem might have caused a company to dedicate too much capital toward the purchase of a mainframe rather than a combined approach with minis and a smaller mainframe for about the same cost but with more utility.

Video teleconferencing is one of the more exciting aspects of office automation since the benefits are obvious and a great relief to the travel weary executive. One video conference between 10 executives who may have had to spend the better part of two days traveling could save a company 20 days of executive time and possibly a few hard dollars in the process when the costs of food, travel and lodging are all added up. Additionally, more people can now attend a meeting or conference since the costs will vary only slightly for the larger attendance.

Calendar scheduling is an important but often overlooked part of office automation that requires a dedicated effort if it is to succeed. The reason calendar scheduling might be a problem is that if a secretary is responsible for maintaining the calendar on the computer and an executive makes appointments in his day timer, conflicts will be numerous and embarrassing. The only way a calendar system can work for scheduling a persons time is if everyone involved maintains the information on the computer in a timely and reliable fashion.

Tickler files or reminders are another form of calendar scheduling where, upon a certain date or time, the computer will notify the person or persons of an event or activity. This type of usage of calendaring makes more sense, as it is meant only to serve as a reminder and thus can have no negative effect and is easier to control.

The above components of office automation are all somewhat related with the exceptions of teleconferencing and video training. All of the functions of word processing, electronic mail, and calendar scheduling can be handled by a general purpose computer system.

Other approaches to implementing office automation direct us toward dedicated word processors and special function terminal networks and elaborate telephone systems with some level of intelligence. Dedicated systems are a risky choice if the premise that a standard has not truly been recognized is correct. Although many stand alone systems have some ability to connect to a mainframe or host computer, this link is usually something less than ideal and is often rather expensive in terms of time, effort, and equipment. In addition, as improvements in technology are announced to the world, the owner of dedicated equipment is oftentimes put into the used equipment business until he can unload the old equipment so that he can purchase the more desirable devices with the latest features.

It is hard to imaging a future office automation system that does not include micro computers (personal computers) since these tools can perform a variety of functions. To date, micros have only done a fair job in the areas of word processing and other simple office automation tasks due to their limited storage capacities, inability to programatically access information on a host computer. I feel that the largest single benefit of the micro computer are the electronic spreadsheets and graphics capabilities that have made them

so popular. These two areas are where micros shine. Even though they do not have the immense storage capability or greater speeds of larger systems, the problem of system degradation involved in large computers is eliminated. The drawback to micros is that they do not readily access the information on the larger system and thus the age old problem of compatibility.

> A few years down the road we will most, likely see a tighter coupling of micros with their larger relatives which will eliminate the problems related to capacity and information accessability. Until then micros will most likely not be considered the total solution to the office automation problem for all but the smallest organizations.

A general purpose system, such as an HP-3000, has far more flexibility in dealing with improved technology since software can be readily changed to accommodate new advances in equipment and also to incorporate new software features as they are formulated. Additionally, having an office automation system on a general purpose computer allows clerical users and managers access to the information which already resides on the system. More advanced word processing systems have the ability not only to handle ordinary text entry and formatting, but also to actually be "programmed" to produce reports based on information to be extracted from date base or data files as part of longer documents. Other system information can be used to produce mailings, dunning letters, reminders and a wide variety of correspondences all automatically without the need to convert or re-key information. Some of the less sophisticated packages available require a complex sequence of steps to extract information from data files and then merge it into text files which is sometimes acceptable. Stand alone systems are even more cumbersome when attempting to deal with large volumes of data and oftentimes require the re-keying of some or even all data to be used.

Another advantage of the general purpose computer approach is that every terminal on a computer system can perform all of the functions of word processing, data processing, calendar scheduling, electronic mail and a variety of other applications. Some of the word processing programs available today require the use of a specific terminal which gives the user the advantage of a more dedicated or "stand alone like" approach. Also included is a higher price tag. The bulk of the word processing programs do not require the use of specific terminals, although there are many that have some restrictions as to the compatability or brand of terminals to make use of all the program features.

A major advantage of having all terminals able to perform the functions of word processing, electronic mail and scheduling as well as the normal data processing is that the cost of implementing such a system is quite low and with very little risk if the terminals to be used are already in place, or are to have dual roles for data processing as well as office automation. In many cases, little or no additional hardware need be purchased to implement a first pass at office automation, just the purchase of one or more software applications. The choice of the software applications for office automation is quite important, not only due to the cost of the software, but more importantly, oriented toward a successful implementation. Spending several thousand dollars for software is trivial when the costs of installing a bunch of applications that are hard to use and don't really do the job can be a major setback in terms of time, training, and morale. An initial failure in this area can also make a second attempt very difficult, if not impossible.

Lets take a look at some of the costs involved for the purchase of different types of solutions.

a. Dedicated hardware.

   Word processing stations normally run between 5 and 10 thousand dollars each with the price gravitating toward or above the high figure when communications capability is added.

   Of the dedicated systems that we have seen none can do all of the office automation functions of word processing, report writing, electronic mail, mass mailings, text editing, and tickler files. So that the comparison to a general purpose computer system is not totally even, although they are getting better all the time.

b. Micro computers.

   Most of the micro computers being purchased for business usage have a starting price around $3,500 to $4,000 but can easily gravitate toward $10,000 with the addition of $2,000 or more in software and $4,000 and up in hardware. Many companies have unwitingly turned their

executives and secretaries into mini DP departments. As time goes on the cost of micro computers will continue to tumble and the performance will increase.

As one might imagine, purchasing $10,000 micro computers for each person that wishes to do a little word processing can get a quite expensive, especially when we consider the drawback of most micros in that they do not easily share peripherals. Micro computers will become a larger part of office automation systems when they learn to communicate with each other, the peripherals and larger systems in a more cost effective way than they currently do. Additionally, a standard operating system and possible a standard architecture may have to evolve to reduce the chances early obsolesence of your chosen micro system.

For those that have made a large commitment to CPM based systems, the MS-DOS "standard" thrust on the market by IBM may cut off the availablity of new software packages in the future. Most of the manuafactures of micro computers and related software are directing their efforts toward the IBM compatable market.

c. Various Software Systems.

Numerous software packages are available from a variety of vendors to provide the functions of word processing, text processing, mailing list processing, electronic mail, report generation, text editing, memo generation, and calendar scheduling. Software packages of this type normally sell for between $2,500 and $10,000 each. If one package were purchased for each function of word processing, electronic mail, editing/text processing, memo generation, reporting, mass mailing and calendar scheduling at an average cost of $5,000 we would be looking at a cost of $35,000 for a complete system. This is just the tip of the iceburg since training and internal support for so many different software systems will be extremely expensive, probably requiring the full time efforts of at least one computer oriented person at an annual salary of up to $35,000 per year. This does not include any overhead associated with the running of the system or the various other costs related to computer usage and ongoing trouble shooting.

d. Integrated Software System

A more rational approach to office automation is to purchase one or more software packages that provide an integrated and uniform solution to the office automation needs of a company. This solution requires a package or packages that are designed to operate under one master menu or control program, allow for a minimum amount of training on each function, and that the training be additive. By additive we mean that each training session uses the previous session as reinforcement and adds to the cohesive approach required for a truly successful implementation. When we refer to an integrated software system we refer to a system where a base set of rules, features, commands, menus and training allows each user to add new functions to their repertoire as they become capable of assimilating the next step of the system implementation. An integrated system is also beneficial, as all system users are dealing with a common set of tools and thus they can share their experiences, discoveries, and knowledge. This reduces the need for dedicated support staff.

We only know of one software system which performs all of the above mentioned functions and this package sells for $9,500 and would most likely require the assistance of a programmer type for initial training and possible ongoing support if all functions are to be implemented. Assuming one fourth the time of the same $35,000 programmer is required, we are looking at $8,750.

What we are pointing out here is that we feel that the cautious approach dictates that a minimum investment be made to procure software that does the "complete job". We do not mean to infer that you should cheap out on software, just purchase the best package or packages that take as much of the office automation concept into account as is feasible with an eye toward the costs of training and implementation. Don't simply buy software from a large vendor because they are large, look at the total picture and you will probably find that the hardware manufacturer that sells software is really leading you into a need to buy more and more hardware and possibly more software to get the complete solution.

Office automation is here to stay, but just as Robert Fulton's paddle boat was a fine solution to a problem, it was not the final solution to the needs of river travelers. A careful study of office automation systems will allow you to see that there are many great sounding ideas but none have really proven themselves. We recommend that your first plunge into office automation be made by adding software which, if worse comes to worst, can be replaced at a relatively small cost.

A point that is often overlooked in any implementation is training. To date I have found no system that is very capable that requires absolutely no training for personnel. Most systems that are "so easy to use that anyone can use them" are so easy because they can do so little. A full function word processor requires training for clerical personnel since the concepts are new and very few people are willing to train themselves. I think that this is the most common failing of those managers who have tried to implement office automation and have failed. These managers will point their employees in the direction of a new word processing system, give them twenty or thirty minutes to figure out how to turn the thing on and then start handing them numerous documents that must be finished before noon.

Training must be handled in a logical fashion with plenty of "free" time for experimenting, mistakes, and extra time for those first several documents. Yes, word processors are designed to make secretaries work easier and faster, but don't expect the results to show up for a week or two and sometimes even longer than that.

The bottom line on a cautious approach is; make use of existing equipment (your HP-3000 and HP terminals), buy a software package that has all the features you will need now and in the future so that you will have an "integrated" solution, and don't overlook training.

I feel very positive about the future of office automation and am glad to be involved in the early stages of such an exciting area of computing. Future developments will bring the cost of hardware down so that we can afford to extend the capabilities of the office automation products to more and more individuals. I hope that this paper will serve to stimulate those thinking about office automation into shopping around and seeing what office automation can do for them. Don't sit back and wait, now is the time to get your feet wet -- just take care not to get in over your head.

About the author:

Mr. Bradford is the president of Bradford Business Systems, Inc. of Laguna Hills, California. Mr. Bradford worked for Hewlett-Packard for 6 years in various capacities ranging from sales to technical support on the HP-9800 series desk top computers, HP-1000, HP-250, HP-300, HP-2000, and HP-3000 computer systems as well as being a data communications specialist for the HP computer products.

Since leaving Hewlett-Packard, Russ has formed his own company which is actively involved with development and marketing of products relating to his experience with Hewlett-Packard products and office automation systems. Immediately after leaving HP Russ performed consulting services concerning a variety of areas, with heavy emphasis on data communications. Since then, Bradford Business Systems has developed products which embrace Russ' feelings as expressed in this paper. His experience in data communications and the various HP systems as well as the time consulting with companies of all sizes has given the products developed by Bradford Business Systems more flexibility and usability in the environments for which they were designed.

If you would like to contact Russ, he can be reached at the following address;

Russ Bradford Bradford Business Systems, Inc. 23195 La Cadena Drive Suite 102 Laguna Hills, CA (7114) 859-4428

RECOVER IT YOURSELF WITH USER LOGGING

by: Diane Weir
Los Alamos National Laboratory

INTRODUCTION

IMAGE logging is a good product that has proved to be an effective and accurate way to save interactive transactions for recovery and audit purposes. There is one shortcoming with the product in that it only logs transactions within the IMAGE domain. Some applications require that KSAM and MPE files be updated in an on-line system. How can these files be recovered? One answer is to use a recoverable program structure that not only posts the interactive transactions, but recovers them as well. The user logging facility is used to store the successful transactions to either tape or disc. This paper will discuss the recoverable program structure and the user logging subsystem.

LOGGING AND RECOVERING TRANSACTIONS

THE RECOVERABLE PROGRAM STRUCTURE

This program structure was developed because there was a need to save all update transactions for an on-line payroll system that used IMAGE, KSAM, and MPE files. A record is written to the log file for every successful update on the file sets. The record logged looks just like the screen's image that caused the update. The on-line system can be run in interactive mode using V/3000 with users at terminals entering their transactions, or in recovery mode. In recovery mode the log file is read instead of the terminal. The edits are re done to insure that the data files were properly restored, then the transactions are posted to the backup copies of the files. If any of the edits fail the program aborts; the recovery is probably being run against the wrong set of files or the date/time parameters are wrong. The log file contains before and after images of the screen for audit reporting purposes; for recovery only the after images are reposted on the change transactions. The deletes contain the before image of the screen prior to the deletion, the adds, the after image.

THE USER LOG RECORD FORMATS

Although the logging subsystem uses several record types, the records to concern yourself with are the user records, coded two and seven. The first nine words of the 128 word record are reserved for the logging subsystem. The first two words contain the record number, the next, the checksum. The fourth word is important in that it contains the record code. Words five through seven contain the date and time. The eighth word holds the log id with the ninth word used to hold the length of the user area. The user area follows for the next 119 words. Records are added to the log file via the WRITELOG intrinsic. The file is in 128 word ASCII format that, through the use of the length parameter, allows transactions of various lengths to be logged. If the write to the log file has a user area of 119 words or less, the transaction will physically be placed into one 2-record. If the 119 word limitation is exceeded, as many 7-records as needed to complete the operation are written. For example, if the length of the user area is 408 words, one will see one 2-record and three 7-records on the log for the request. This gives the user the flexibility needed for various uses of the log files.

When designing the application the 2-record was defined to contain certain control and audit information. In COBOL syntax the log records were defined as follows.

```
01  LOG-RECD.
    05  LOG-SYSTEM-AREA.
        10  FILLER                 PIC X(6).
        10  LOG-REC-CD             PIC 9(4)    COMP.
        10  FILLER                 PIC X(10).
    05  LOG-USER-AREA.
        10  LOG-USER               PIC X(8).
        10  LOG-SCREEN-CD          PIC XXX.
        10  LOG-BEF-AFT            PIC X.
        10  LOG-ACTION             PIC X.
        10  LOG-DATE               PIC S9(7)   COMP-3.
        10  LOG-TIME               PIC S9(5)   COMP-3.
        10  LOG-SCREEN-IMAGE       PIC X(218).
    05  LOG-CONT-AREA  REDEFINES  LOG-USER-AREA.
        10  LOG-CONT-SCREEN        PIC X(238).
```

The LOG-SYSTEM-AREA is the nine word area reserved by the user logging facility. The LOG-USER-AREA is the definition of the 2-record. It contains the user name obtained from the WHO intrinsic, the application's screen code, a before/after code, the screen's action code, the posting date and time, followed by the first 218 characters of data. If a continuation record was needed the LOG-CONT-SCREEN contains the 7-record's data. The before/after code is used primarily for audit reporting. An add transaction will contain only the after record; a delete, the before record. A change transaction will reflect both the before and after states. The screen image is remainder of the data written to the log file.

THE MAIN PROGRAM

The on-line system contains a main menu program that prompts for passwords, opens the database, terminal, and formfile, and displays the menu. The sub programs actually update the files. The menu's function is to control the flow between the subprograms. The menu can be executed in interactive or recovery mode. The main program knows if recovery or interactive mode is desired via the use of run-time parameters. SW1 in COBOL was used if recovery mode was needed. Thus to run the program interactively, simply ;RUN PAO001, to run for recovery, :RUN PAO001;"PARM=%40000". The logic of the two modes is illustrated as follows.

INTERACTIVE MODE

```
Turn echo off and ask for passwords
Open the database
Turn echo back on
Call the WHO intrinsic to find the
    user's name.
Gain access to the logging facility via
    OPENLOG
Open the formfile.
Open the terminal

READ-LOOP
Display the menu screen
Read the menu
Call the subprogram to service the
    request
    -or-
Go to the EXIT-ROUTINE if F8 was
    pressed.
Go to READ-LOOP
```

RECOVERY MODE

```
Read the EDITOR file that has the from
    date/time and to date/time for the
    recovery process.
FOPEN the log file specifying an old,
    permanent file, opened for exclusive
    access.

READ-LOOP
Add 1 to the record number.
Call FREADDIR to read the file sequen-
    tially.
If the log record code is not a 2, skip
    the record.
Test the date/time in the log record to
    see that it fits the recovery para-
    meters.
If the log's time is less than the
    recovery time, read the next record.
If the log's time is greater than the
    recovery time, go to EXIT-ROUTINE.
Subtract 1 from the record number of
```

```
                                    the log file
                                  Call the appropriate subprogram.
                                  When the subprogram returns test the
                                    returning screen code for the end-
                                    -of-file flag.  If it is not set,
                                    subtract 1 from the record number and
                                    go to READ-LOOP.  If it is set go to
                                    EXIT-ROUTINE.

EXIT-ROUTINE                      EXIT-ROUTINE
    Close the terminal and formfile     Close the log file via FCLOSE
    Close the database and other files  Close the database and other files
    Terminate the access to the logging Stop run.
      facility with CLOSELOG.
  Stop run.
```

When a system failure occurs in the middle of the day, the operator must first restore the files from the latest full and partial backup sets. Then the date and time of the last backup tape is entered into an EDITOR file along with the date and time of the system failure. This delimits the recovery process to the time parameters saved in the log file's 2-record. The operator then stops the logging process with the :LOG console command.

The database is opened for exclusive access while recovery is running. This insures that no processes are using the database until recovery is completed. The log file is also opened in exclusive mode as an extra safety measure. A stop of the logging system forces all buffers to be flushed to their media so the log file should be as complete as possible. If logging was not stopped prior to recovery the exlusive open of the log file will fail. The log files is then read using the FREADDIR call because the subprograms need to know where to begin processing on the log file. The record number is reduced by one prior to calling the subprogram so the subprogram can add one to the record number before reading the log file. This keeps the subprogram's read loop consistent.

THE SUBPROGRAM

The subprogram's structure is described below. In interactive mode the screen is read and the data is edited. If the edits do not detect errors the database and other files are updated. For delete and add transactions, the screen image is added to the log file via WRITELOG directly from the screen image in working-storage. On change transactions, the "before" screen is re built and written to the log file before the updated screen image is logged. By comparing the "before" screen to the "after" screen on the audit report the changes can be isolated. The logic for the subprograms is outlined below.

INTERACTIVE MODE

```
READ-LOOP
Read screen (VREADFIELDS,
             VFIELDEDITS,
             VGETBUFFER)

Edit the transaction
   (If errors perform VSETERROR
      then go to READ-LOOP)

Update the datasets and other files
```

RECOVERY MODE

```
READ-LOOP
Add 1 to record number, then FREADDIR
    the log file.
Bypass any records whose code is not a 2
Test the "to date/time" against the rec-
    overy parameters.  If the time has
    expired or the end of file is found
    return to the menu.
See if the screen belongs to this
    subprogram, if not return to the
    main program.
See if the screen was too large to fit
    into one log record.  If so, cont-
    inue reading until the entire screen
    is reassembled.

Edit the transaction
    (If errors perform VSETERROR)

Update the datasets and other files
```

```
If add or delete, write screen to log
If change, build "before" screen
          write it to the log then
          log the "after" screen.

Initialize screen for next transaction.

Go to READ-LOOP                          Go to READ-LOOP
```

VSETERROR ROUTINE

If program is in interactive mode, call VSETERROR for the field, else abort.

In recovery mode, the routine to edit the screen's data and update the files is the same routine performed when recovery is run. There are not two separate programs to maintain when changes occur to the edit or update criteria. The same subprogram that updates the datasets, KSAM, and MPE files interactively also recovers those transactions. A word of caution. Since the edit routines are performed for the recovery to insure data integrety, any alterations to the edit criteria or the screen layout should be preceeded by s :STORE of the data- bases and other files updated by the system.

There is no need to delimit the logical transactions by special records written to the log file. IMAGE logging delimits transactions by DBBEGIN and DBEND calls. This is to prevent any incomplete updates from occurring. This is not needed in this type of structure. The screen is the logical transaction. One screen may update a variety of files but since the screen is being recovered instead of the records, special transaction delimiting records become unneces sary.

THE COMMON AREA

The common area of the on-line system contains the V/3000 area, the database name, and the data needed for the logging and recovery. PP-USER comes from a call to WHO to determine the user's name. PP-LOG-INDEX is the log index returned from the call to OPENLOG. LOG-FILE-NUM is the file number for the log file when run for recovery; it is required to FREADDIR the log file. The PP-RECOVER-FLG indicates the mode, recovery or interactive, to the subprograms. PP-REC-NUM indicates where recovery is to begin on the log file. PP-SCREEN-CODE tells the main program the next screen to process or whether the subprogram reached the end of file or the time limit was exceeded. The recovery is terminated when log file ends or the log record's date and time exceed PP-RECOVER-TO-TIME.

```
01  COMMON-AREA.
    05  PP-D-BASE         PIC X(8).
    05  PP-USER           PIC X(8).
    05  PP-LOG-INDEX      PIC S9(9)  COMP.
    05  PP-REVCOVER-FLG PIC S9(4)  COMP.
    05  PP-REC-NUM        PIC S9(9)  COMP.
    05  PP-SCREEN-CD      PIC X(4).
    05  PP-EOF-FLAG REDEFINES PP-SCREEN-CD
                          PIC X(4).
    05  PP-RECOVER-TO-TIME.
            10  PP-T-DATE     PIC S9(6).
            10  PP-T-TIME     PIC S9(4).
        05  LOG-FILE-NUM      PIC S9(4)  COMP.
        05  V-COM-AREA        PIC X(102).
```

TESTING CONSIDERATIONS

How is testing conducted on a logging system? When testing occurs against a test database, the transactions should not be logged to the production log file. Instead the transactions are logged to a test log file. The log file identifier in main program is altered prior to the OPENLOG call. Also, the recovery should be tested if the screen layout was altered. This mandates that the FOPEN of the log file in recovery mode use the file name of the test log file, not the production log file. This can be accomplished through a file equation. Again the run parameter, SW2, was used to indicate whether testing was occurring. To test interactively one would :RUN PAO001;PARM=%20000, for testing recovery, :RUN PAO001;PARM=%60000.

THE USER LOGGING SUBSYSTEM

THE LOGGING PROCESS

The user logging subsystem allows for one shared file buffer per logging process regardless of the number of users accessing the log file. A write is performed on a log file via the WRITELOG intrinsic. One may log to either tape or disc. For disc logging, the log entries are loaded into the buffer area of the logging data segment. The records are written to disc when the buffer area becomes full or when certain intrinsics such as FLUSHLOG, BEGINLOG, or ENDLOG are called. Tape logging actually writes the log buffer to disc for a later transfer to tape. Transfers to tape occur simultaneously with writes to the disc log file because the two steps are controlled by separate processes. The reason for the two processes is so that the process that loads the buffer to disc can continue without interruption. The process that writes the transaction to tape can pause while a reel rewinds and another is mounted. This gives the logging process the capability to continue without interruption at the end of a tape volume.

IMAGE uses the user logging subsystem to record updates to the databses where logging is enabled. The user logging facility was written by a team in the MPE group to provide the framework for IMAGE logging. IMAGE records physical records updated by DBPUT, DBUPDATE, and DBDELETE calls if the logging on that database is active.

GETTING STARTED WITH USER LOGGING

This section will deal with the commands and utilities used for the logging subsystem. All users accessing the user logging facility need to have logging, or LG, capability. The system manager needs to allocate LG capability to the account, then the account manager can allocate LG capability to himself and to the users accessing the interactive logging system.

```
:HELLO MANAGER.SYS
:ALTACCT PAYROLL;CAP=AM,AL,GL,OP,ND,SF,IA,BA,LG
:HELLO MGR.PAYROLL
:ALTUSER MGR;CAP=AM,AL,GL,OP,ND,SF,IA,BA,LG
:ALTUSER SALLY;CAP=ND,SF,IA,BA,LG
```

Estimate the size of the log files. They should be large enough to contain at least one day's worth of transactions. You may want to set the file size large enough to hold a week's worth of transactions if weekly audit reporting from the log file is desired. Build the log file with a record length of 128 words and a file code of LOG. Decide on a log identifier (log id). The log id is your link to the logging subsystem. Use the :GETLOG command to associate the log file with the log id, to tell the subsystem where logging is to occur, and to assign a password to the logging access. The password is not mandatory.

```
:BUILD PAD100;REC=128,5,F,ASCII;DISC=15000,16;CODE=LOG
:GETLOG PADLOG;LOG=PAD100,DISC;PASS=
```

OPERATIONAL CONSIDERATIONS

The command to actually start the logging process is the :LOG console com mand. There is a problem with the console commands in that one must have been allowed the command in order to issue it from somewhere other than the console. The contributed utility ALLOWME will grant console command capability to users other than the owner of the console. The account manager was allowed the LOG command. OPERATOR.SYS was allowed both the LIMIT and LOG commands so that these can be controlled by the batch job running the SYSDUMPS.

```
:RUN ALLOWME.UTIL.SYS
Allowme Utility   V0.0   19 January 80
MGR.PAYROLL;COMMANDS=LOG
END OF PROGRAM
```

The jobstream for the SYSDUMPS sets the LIMIT to zero then stops the logging processes before a full or partial SYSDUMP. This allows the log file to be saved on the backup tape. After the SYSDUMP has completed, the jobstreams restart the logging processes and raise the limits back to normal. It is useful for the account manager to have access to the :LOG command and OP capability so that the logging process can be stopped and all files, including databases, can be stored prior to clearing the log file. OP capability allows a user to store a database without needing dangerous PM capability.

```
!JOB PARTIAL,OPERATOR/OPPASS.SYS/SYSPASS
!RUN ALLOWME.UTIL.SYS
!LIMIT 0,0
!CONTINUE
!LOG PADLOG,STOP
!FILE TP;DEV=7
!FILE LP;DEV=LP
!SYSDUMP *TP,*LP

10/20/83

Y
!LIMIT 2,16
!LOG PADLOG,RESTART
!EOJ
```

The system manager might need to alter the system configuration for the logging to work on your application. In the system table section of the SYSDUMP dialog, the manager defines the maximum number of logging processes allowed on the system at any one time and the maximum number of users per logging process. The system manager manual recommends 20 for both of these parameters but that might not be enough. When assigning these numbers remember that any IMAGE log ging performed on the system needs to be taken into account also.

One last operational consideration for the user logging facility is the OPERATOR.SYS startup procedure. Some shops stream a job and others use a UDC file. In the logon UDC for the console, the logging processes are restarted via the :LOG command.

```
STARTUP
OPTIONS LOGON,LIST
ALLOW OPERATOR.SYS;COMMANDS=CONSOLE
LOG PADLOG,RESTART
HEADOFF 6
STREAMS 10
JOBFENCE 4
OUTFENCE 4
STARTSPOOL LP
ALLOCATE EDITOR.PUB
 (etc) ...
```

LOGGING COMMANDS

There are other logging commands that help one use the facility. :LISTLOG lists the active log identifiers on the system and their creators. :RELLOG del etes log identifiers from the user logging facility. :ALTLOG changes certain characteristics of the log id such as the log file name, the log destination, or or the logging password. :SHOWLOGSTATUS displays the status of all currently active log files. When the CIPER MIT was installed, the log identifiers were corrupted. The fix was to delete the bad log ids with :RELLOG and add the good ones back with :GETLOG. Fortunately, the log files were intact, just the identifiers were corrupted.

```
:LISTLOG        <<lists active log identifiers>>

    LOGID               CREATOR              LOGFILE
    PADLOG              MGR.PAYROLL          PAD100.PUB.PAYROLL

:ALTLOG PADLOG;LOG=PAD100,TAPE      <<changes log characteristics>>
:

:SHOWLOGSTATUS              <<status display of active log processes>>

     LOGID          USERS     STATE        RECORDS
     PADLOG           0       INACT         225
```

```
:RELLOG PADLOG          <<deletes a log identifier>>
:
```

THE LOG RECORDS AND THEIR FORMATS

There are nine record types in a log file. The format of the log records vary depending on the record type. Record type one is the openlog record. It is generated whenever a user accesses a logging system via the OPENLOG call. The three-record is the closelog record, generated when a user executes the CLOSELOG intrinsic. There is a start or restart record, code six. Records coded four and five are the transaction header and trailer record generated by the BEGINLOG and ENDLOG intrinsics. IMAGE uses these for DBBEGIN and DBEND calls. BEGINLOG and ENDLOG cause the logging buffer to be flushed to disc; so do DBBEGIN and DBEND calls. The nine record is a crash marker. When logging is restarted after a system failure while logging was active, recovery occurs on the log files. The crash marker tells the user logging subsystem where the crash occurred so it can recover itself. The user records, code two and seven, were discussed in the first section.

INTRINSICS USED IN LOGGING

A write call to the logging subsystem uses a mode parameter. This parameter tells the logging system which action to take if the buffer becomes full prior to the write request. Mode one functions similar to no-wait I/O; the process continues after passing the request to the logging subsystem. Mode zero forces the process to wait until the logging system has processed the write to the log file. If the buffer is full and the write to the log file in mode zero, the buffer is written to disc and cleared. The entry is placed into the buffer, before control is returned to the calling program. The mode is also used for other calls to the logging system and operates in the same fashion.

Access to the logging facility is obtained via the OPENLOG intrinsic. The format is OPENLOG index, logid, password, mode, status. The index returned is used on subsequent calls to WRITELOG and CLOSELOG. The logid contains the log identifier, the password, the logging password. The mode indicates the wait request as discussed above. The status will contain error codes if the OPENLOG call failed.

The WRITELOG intrinsic format is WRITELOG index, data, length, mode, status. The index is the same index returned from the OPENLOG call. The data is the user data to be written to the user area of the log record. The length is the size of the data being passed. Again the mode is the wait/no-wait request.

The CLOSELOG intrinsic is used to stop access to the logging facility. Its Its format is CLOSELOG index, mode, status. Index, mode, and status are the same as for the WRITELOG intrinsic previously described.

CONCLUSION

There are various methods available to users to recover lost interactive transactions. The method previously described is one way to approach the task. IMAGE logging is probably preferred since the programmer does not have to be involved with recovery. Unfortunately, IMAGE logging is not always the answer. There are files outside the IMAGE domain, KSAM and MPE files, that are updated via interactive programs that also need to be recovered. The user logging fac ility is an efficient answer to save those transactions that are critical to the application. The recoverable program structure described may be a useful tech nique since the chances of inconsistent results between two separate posting programs are eliminated. There is extra time required to develop and maintain the self-recovering programs, but the time is probably less than having one program post interactively and another post for recovery. There is a better chance of data consistency if one program does all the posting, be it inter active or recovery.

BIOGRAPHICAL SKETCH of Diane Weir

I am currently employed by Los Alamos National Laboratory as a staff member using DEC equipment. Formerly, I was the data processing manager for a heavy construction firm using the HP 3000 for three years. It was in this capacity that this recovery technique and the User Logging subsystem was developed. I began in data processing in 1970 and have worked on IBM, Burroughs, Hewlett-Packard, and Digital hardware. I have a Bachelors of Business Administration from the University of Texas at El Paso.

HP 3000 - Sizing and Performance
More Machine for Your Dollar, or "Where Did All My Hardware Go?"

by John S. Parkinson

1. Introduction - Why Sizing and Performance?

One of the commonest and most widely propounded theories of the computer industry over the last ten years (probably longer) concerns the falling real cost of computer hardware. We are always being told that any day "soon", hardware will be so cheap that it will be "given away" with application software. If this was true, there would be no need to worry about how much hardware was needed to process a given (and always increasing) workload, since a grateful manufacturer would simply present you with more equipment, whenever the need arose!

You have probably found it difficult to identify which particular hardware suppliers are taking this approach, and how many of those who do (if any) are still in business. As an HP 3000 user, you will have been attracted to Hewlett Packard computers by their ability to meet your particular needs, and to deliver the required performance at the right price. In doing so you will have already discovered the first principle on which this paper is based.

> TANATAAFL - or "There aint no such thing as a free lunch"

Unless you have an unlimited source of funds for acquiring computer equipment (unlikely but possible), a decision to purchase or upgrade a system requires justification. Unless you are very lucky, you will only be allowed to buy as much equipment as you need to process your workload. You therefore need to get your configuration right, and this is where sizing and performance assessment comes in.

HP, who are as honest a group of computer sales people as you are likely to meet, will be quite happy to tell you what you need, but how do you assess whether their advice is adequate or correct? This paper gives you some guidelines, and suggest some other profitable sources of information and advice.

One final point. This paper is being written in November 1983 and hence refers to the state of play as of now. I have tried to include what I know about MPE-V, disc cache etc, but the delays and changes in the release of MPE-V make it difficult to be exact. By the time you see this, some of you MIGHT have MPE-V in use. If so you will know if I got it right (or even if HP did). So please treat the sections that deal with possible developments with caution. Also, be aware that most of the material in this paper is a simplification of a complex process. I have included general examples wherever possible, but they ARE generalisations and may not apply to your particular site. If you do have a problem, ask some one; HP; another user or the user group. Do go to user group meetings. I doubt if your problem is unique.

2. Workload Categorisation

The first place to start a sizing and performance exercise is with the work you expect your computer to process. This may seen absurdly obvious, but I have seen a lot of sites where what a system is actually used for varies quite a lot from its stated raison d'etre.

Minicomputers like the HP 3000 are extremely powerful general purpose tools that are DESIGNED to handle a wide variety of possible tasks. It is even possible to do SOME of these tasks AT THE SAME TIME. If, however, you expect to be able to do ALL of the possible

kinds of work simultaneously, you are either going to be disappointed or to buy a lot more hardware than you thought. Ten years ago there was no problem, since computer systems only did one kind of job - batch processing. Thanks to progress, your HP3000 could now be used for some or all of the following:

Batch Processing (this one will never go away) Online transaction processing Online data capture Electronic mail Word processing Communicating with other computers Telex management Document design and printing

and last but not least, software development and maintenance.

This is a lot to ask of a computer, and to enable all these facilites, the designers have to compromise so that, although each will work, none works as well as it would alone. Since different classes of task have different processing requirements, no single architecture will satisfy all equally well, and anyway (whisper it) some of the system software designers were better than others(!) and hence some applications work better than others too.

So the first step we need to take is to figure out what our system is supposed to do. Recognise that this is only a CURRENT viewpoint, and will change as we progress, so also expect to keep an eye on what the system is actually doing from time to time. If we can use the system for several different jobs with the same characteristics, we can tune the machine to its optimum for these jobs and so get more work through a given configuration. If the job mix is very varied, we can review the desireability of processing it all on a single system, and so on. If, however, we don't know what the system is to do (or doing) none of this is possible.

Workload categorisation is not easy, especially for users with no previous computer systems experience, but is worth doing, even if you ignore the rest of this paper. If you don't feel able to do it yourself, get some outside help. You will almost always save money in the medium to long term. At the very least, write down a list of applications and see which category from the list shown above they fit into.

Once general word of advice. The system development tools on the HP3000 are very good, and tempting to use. However, interactive software development eats up system resources at an alarming rate, and all too often production work suffers. If you are a software house, this is no problem, but if you are really there to process an MRP or Accounting application, keep the programmers off the machine! It will often be cheaper to buy them one of their own, than to provide a configuration that will run both kinds of work side by side without affecting response and run times.

Also remember to allow for MPE and other system products. They all need disc space, memory and CPU cycles and the more facilities you use, the more resources MPE will take up. This is important, because there is relatively little you can do about it.

3. Throughput Assessment

Once you have decided what kinds of work are to be run on your system, the next step is to estimate how much of each kind of work needs to be processed. This also seems rather obvious, but it is important to produce estimates that reflect factors relevant to the overall system performance, not just the computer equipment. Various levels of sophistication can be employed in generating estimates, from simple counts to complex stochastic models, depending on how critical a factor the pattern of throughput actually is. You should not produce a model that is more complex than necessary, nor ignore the estimatation process altogether. Quite a good idea is to generate an 'average transaction' for each type of workload and use this, but remember that peak loads can often be many times the average value, and that some processing jobs will be time critical. Once again, if in doubt, get some help.

The throughput assessment tells you several important things. First of all, it will give you an estimate of the MINIMUM number of terminals (VDUs and printers) needed to process the interactive workload. Note that this may be less than the configuration requirement, since it takes no account of location or convenience factors, and is often based on fairly crude estimates of workflow, but it is a lower bound, and determines which series of 3000 computer will suit you.

Second, it will provide an estimate of how much file I/O capacity you will need. At this point you should also estimate how large (and how many) your files will be, and decide on their structure (IMAGE and KSAM files have variable 'overheads' dependant on the complexity of the data structures in use). This will determine your options for disc configuration, and thus how many access paths to the discs.

Finally, you will get some idea of how much memory the system will need. We will look at this later, but in general, always get as much as you can afford. Details of other peripherals (system printers, magnetic tape drives etc) also come out at this stage, but are relatively easy to determine with some simple arithmetic.

If you expect to run a lot of batch work as well as (or alongside) your interactive load, you have probably made the wrong choice of computer. However, this needs to be estimated here too, as it will contributed to memory, CPU and disc usage. Since there is no terminal wait time on batch jobs, they can use up a lot of file I/O, even if they are put in a lower priority queue.

4. System Limits

Once we have a picture of the type and volume of processing that our system needs to do, we can look at the various types of HP3000, and see which best fits our needs. The first thing we need to be aware of is that each model in the range has certain limits, some imposed by the architecture and some by HP. We will start with some 'overall' limits and work inwards. Configuring an HP3000 gets more complicated all the time. Five years ago, HP gave you a simple diagram on which you could tick boxes to decide what components you needed, and it was clear 'how many of what' could be added. Now you have a four page long list of questions to fill in, and even this does not cover all the possibilities. This growth in complexity has been, in part, the result of the move from the old Series II/III machines to the HP-IB bus architecture, and the attempt to use "standard" components for interfacing to the bus. all subsequent remarks refer to the HP-IB (and later) machines, and do not to the earlier SIO bus.

The use of the HP-IB introduces our first real limit. The maximum clock rate for the bus is c. 9.5MHz, which allows data transfers (on the IMB) at a maximum rate of 3 Mbyte/sec. The GIC maximum is less than this, at 1Mbyte/sec. To maintain this rate, the bus length must be short, so that the electrical properties of the backplane and cables used do not degrade performance. For high speed devices, therefore, the maximum practical bus length is c. 6m. We will return to this speed limit in later sections.

The other most critical limitations are as follows:

Power Supply - Each board plugged into the backplane draws power from the system processor unit power supply. Even if you are careful to use a sequenced power up on your system this is a major limit. It seems that whoever designs the power supply for HP was never told that anyone might want to actually fill up the card cage, so there was no need to allow for this possibility in design!

In practice, you can juggle the total power consumption loading a good deal more than HP allow. The configuration manual rules tell you what is formally allowed, but friendly CE0 managers may allow some deviations. (If not, you could consider doing your own maintenance!).

Card Cage - This is what determines (PSU allowing) how many boards can be plugged into the HP-IB backplane. Slots are NOT interchangeable, however, because some boards need to be linked together by ribbon cables and/or have hard wired addresses within the bus controller program. This means that some slots cannot be used. On the series 4X, for instance, there are 17 slots for memory boards (based on the original 16 x 256KB boards + 1 add on controller). With 1024KB boards you only need 4 of these to get the 4Mbyte system maximum, but the remaining 13 cant be used for anything else! With a rumoured 4 Mbyte board (using 256K RAM chips) rumoured for early 1985, this situation will be even more riduculous!

The remaining 26 slots are for ADCC's (or SIB/ATP's), GIC's, INP's etc. However, even here you don't have a free hand. Originally 15 ADCC's could be installed (total 60 ports). With the advent of the ATP, this could be reduced to 7 boards (one ADCC is required plus one SIB and 5 ATP's) for a total of 64 ports and 8 free slots. However not all these can be reused, because of PSU limits!

Once again, we need the configuration rules to tell what is allowed and what is not.

Junction Panels - Each I/O card communicates with the outside world via a cable/connector which HP like to look tidy. This in effect means that each machine uses a different cable (well they have different part nos) which is NOT transferable, even if the board is! The CEO doesn't like cables hanging out of the back of machines (unless they're series 39/40/42!). So you need the correct version, plus a junction panel to screw it to. When you run out of junction panel space - thats it; no more connections allowed.

MPE Kernel - Ever since the 3000 range 'mushroomed' from the old series III, there have been two parts to MPE. This is necessary, since at the hardware level, the machines are radically different. However, HP also chose to incorporate other limits, which determine, for instance, the memory and channel address limits and DRT liits for a particular series machine. I don't know why they did this but there you are. So, even if you COULD put 16 1Mbyte boards in a series 4X it wouldn't do you any good since the 4X version of MPE wont talk to an address above 4Mbytes. If you think this is daft, please write and complain. It might do some good.

System Tables - There are LOTS of limits in this bit. If MPE's to talk to a configured device, there must be a DRT entry for that device. The size of the DRT therefore limits the maximum configuration In practice, c150 entries is the most available under MPE IV which in most cases is more than can actually be used, anyway. Some other interesting limits are:

- Code Segment Table Entries (192)
- Process Identification numbers (256)
- File system directory entries
- Open files per process and overall
- Buffer space allocation

If you come REALLY close to some of these limits, you can run into others as well. However, MPEV will change all

this, so we wont dwell on it here. See below for the latest news.

Program Size - The HP3000 is a 16 bit computer. The maximum directly addressable offset is therefore 64K. Since opcodes only use a maximum of 15 bits for an address, this limits a CODE segment to 32 Kbytes, although data segments can be full 64K. Since you can have up to 64 code segments in a program, the maximum program code size is 2 Mbytes (note: you can actually build programs bigger than this, but not load them). Note also that the limit to VIRTUAL memory is 60,000 sectors of disc, i.e. 15 Mbytes, and that this is less than twice the maximum real memory size on a model 6X.

HP Support - In the last resort, HP will only maintain and support a system configuration that they have actually tried out and therefore know will work. How far you can push this depends on the attitude of your local CEO and what you want to do. For most sites, it is best to stay within the published guidelines and ask politely it you want to do something new.

5. Critical Resource Assessment

We now have most of the information needed to begin looking for particular problems. This is where the going gets a bit more difficult, as it is often not obvious which problem we should be looking at, and solving the wrong problem is all too easy. For our purposes we can divide the computer up into a small number of "components", the size or capacity of which might be a problem. In every system, there will be at least one of these that we don't have enough of. This is the CRITICAL MINIMUM resource, and unless we do something about it, other changes will not improve performance or throughput. The trick is to find it! Here are places that it is worth looking in detail.

CPU - The current CPU's in the 4X and 6X series machines are fairly fast (0.4 mip and 1.0 mip) so it is unlikely that these will cause bottlenecks. Remember, however, that effetive use of the disc cache facility will use 20 - 30% of the available CPU cycles, and that I/O interrupts on the CPU can also degrade performance. Note also that if the capacity required for disc caching is NOT available, then performance will DEGRADE!

Main Memory - This is a hot favourite for problems. You get quite a lot of main memory on current systems (0.5 Mbytes to 8 Mbytes) BECAUSE YOU NEED IT In fact, I don't know why HP sells ANY HP3000 with less than 1Mbyte of memory, since 0.5Mbyte systems run out of steam under virtually any real loading. The 6X Machines also have an 8K high speed cache (why not 64K?) for high speed access. Access time is not really the issue, however, since a system with, say 60 users has a need for a large memory size rather than raw speed. The need to use disc I/O to retrieve virtual memory segments slows things down a LOT (ten to fifty thousand times slower)

so the more active code and data in real memory the better. I have seen 4X based systems where 70% of all disc I/O was virtual memory swaps. Disc cache WILL NOT HELP this, since it needs large chunks of 'spare' memory to hold files, not programs and data, and if there is not enough of this (say 25% min) performance will degrade.

Note that MPE itself uses a good deal of memory (c. 180 Kbytes minimum to c. 850 Kbytes max. I have never seen a 3X or 4X system exceed this figure, although it clearly could do so in theory. 6X system can use up to c. 1.5 Mbytes).

Disc I/O - This is a complicated area. Disc I/Os comprise three main types:

1 - Virtual memory swaps (see above)

2 - System I/O, covering MPE activity, spooling, logging (system and database) directory management etc. Much of this is "invisable" (i.e. not directly caused by applications code) and so gets forgotten. It can add up to 30% to the total so it needs to be included. Disc cache (memory permitting) could help here by holding log files etc in memory but Note also that much of this activity is a WRITE to disc rather than a read, so cache may not help after all!

3 - Application program I/O to data files (IMAGE, KSAM etc). This should be the biggest volume of I/O and is the area that disc cache is designed to help. To do so, your disc activity needs to be 75% reads (or better). In many online applications, this is a good bet, given the overheads involved in directory reads and data structures, and that enquiries and amendment transactions all involve more reads than writes. However some applications, (e.g. sorts, file creation, data capture) may be 50 - 100% writes, and disc cache wont help much here.

Disc I/O efficiency also depends on physical configuration factors. The nearer the ratio of controllers to drives, is to one, the more paths to the data exist, so I/O can be in parallel. The more spindles per volume of data, the less head contention will be present. Multiple GIC's further increase the level of parallelism available. The secret of this is to use as much as possible of the available band width. As was mentioned above, we have 3Mbyte/sec on an IMB. Since disc I/O is fast (1Mbyte/sec in burst mode) we can use a lot of the available band width if we arrange transfers efficiently. In the past, the BEST we could do was c. 30 I/Os per sec, and most systems achieved less than this in

practice. Now, with the new CS80 discs the 4X and 6X machines can delivery 60 - 150+ I/Os sec, but they still need to be utilised to be effective.

The trend is also to provide more disc space per spindle, at a lower cost per bit stored. This is OK in theory, but with more tracks to search, a 400 Mbyte disc had a WORSE average performance than the old 7920/25 discs This has now been put right, so that all the discs (791X, 792X, 793X) have more or less the same performance on a per drive basis. How they perform on a particular system depends on lots of factors, like buffer size, file organisation, use of private volumes etc.

Finally it is worth mentioning that HP are introducing Rotational Position Sensing (RPS) on their CS80 disc drivrs. This senses when the right sector will come along and be under the Read/Write heads and frees up the GIC while waiting. Although this wait is "small" (c. 10 ms equal to the average rotational latency), the GIC can do a lot of work in the time if it has to service several discs.

Terminal I/O

Terminals and other slow devices work so much slower than everything else that you might think they are no problem. Usually this is so but there are one or two things to remember.

ADCC's generate a CPU interrupt on every 'block' transfer from an attached device. This is bad news if your CPU is heavily loaded but is usually only a problm on 3X series machines. This is why block mode applications use less CPU than character mode. However a block mode transfer can use up lots of buffer space both in main memory and on the ADCC board. We have found that using V/3000 screens at 9600 bps on an ADCC can result in data loss if all 4 channels run like this and the system is heavily loaded. ATP boards don't have this problem. They do DMA transfers and don't involve the CPU.

Other Things -

Like system backups to tape. In theory you could put your system log files on tape (does ANYBODY do this?) But in practice you need to use STORE or SYSDUMP now and again. If you have a lot of disc files that change regularly, this can take a LONG time, even if you pay a LOT of money for a 7976 tape deck. Help is on the way in the form of a dual mode MT with fast streaming for backup and start/stop for other uses. (7974). By the time you read this you should know how much it will cost.

System printing via the spooler is another problem. Every spooled printing operation uses disc transfers and GIC bandwidth. Still, you could always buy a laser printer.

In practice, one of CPU, Main Memory and Disc I/O will be the critical resource. When you find out which it is, you can try to do something about it. We will look at tools for this later.

6. Complications

So far our approach has been based on a relatively simple single system example. Real life can be much more complicated. Networking, RJE, IML, Graphics, the list of possible complications increases all the time. The biggest complication however is supplied by the applications software being used, and the way in which it is designed, operated and mixed. The first thing to remember is that individual processing loads to not add up linearly. Two applications that each use 20% of a resource do not, when run together, usualy use 40%. The relationship is closer to exponential. This is because of the myriad of individual queuing interactions that take place within the system. Sorting this out is a job for experts and specialist tools. In general, however, a small number of applications can support more users than a larger number.

The efficient design of applications makes a big difference too, but not as much as some people think. Most problems come from a few design factors, such as:

- database locking strategies segmentation
- database design trying to be too clever
-

System level software can also cause unexpected complications, as anyone installing HPWORD will have discovered. At least when your own applications are compiled and prepared, you get some clues on how much resource will be needed to run them, and can control segmentation. With HP's own software, this is all outside your control. **Sometimes this shows!**

If you want an excellent view of how to assess the basics of performance/design criteria, try Jim May's "Programming for Performance" in the proceedings of the 1982 IUG conference in San Antonio. There is plenty of other published advice on this around, so read INTERACT regularly and write or call the people who contribute articles. Even better, write and describe your problems, you might get published!

7. Performance

Now that we have identified all the main components of sizing and effective use of resources, we can begin to assess what our system will do. I do not intend to go into a lot of detail here, as much has already been written on performance and there are bound to be other contributions during the meeting that will address this directly. However here are some general observations reflecting important trends.

- Anything you do on the HP3000 will go faster if written in SPL. HOWEVER, it wont necessarily go much faster, and in SPL you have to do it RIGHT to get the benefits. There are also lots of ways to do it wrong, and make things worse for everyone else on the system as well. So stick to COBOL (or whatever) where the chances of REALLY bad code effects are reduced.

- If you use a code generator or applications development system, don't expect to get as resource efficient applications as you would if you wrote code. Remember TANSTAAL. What the so called fourth generation approaches give you is applications that arrive faster, not run faster. We have run some tests on several of the available systems and found that, used at their simplest, they are 30%-50% less resource efficient than good COBOL code, while at their most complex, they get to within 10% of the equivalent COBOL. Of course you can go just as badly wrong in application design using these systems as with traditional methods.

- If you have a memory bound system, avoid the use of block mode, which uses lots of memory for buffers. If you still want to run V/3000 applications (you probably will), use something like PREVIEW from TYMLABS to convert to character mode. Remember, however, that ADCC based systems will generate many more CPU interrupts in character mode (TANSTAAFL again), and this may affect disc cache performance.

This is all getting rather complicated. To sort our what is going on on the system, we need some quantative data to identify which resources are over or under utilised, and to direct efforts to the most critical areas. We will look at these next.

8. Tools

All the above is of little use (outside of intellectual exericse) if you have no way to measure what is going on in a real situation. To do this you need some tools. Fortunately there are several available, both from HP and from third parties. Most of them work OK and some of them are useable by non experts. Unfortunately, many sites don't think its worth spending money on them. This is silly, since the TOTAL cost of ALL the available (and usable) tools, plus the cost of help in learning to use them is probably less than $50,000. A good representative selection, including some in the contributed library, probably costs less than $10,000. Most systems costs a LOT more than this, and most sites annual upgrade budget is more than this too. Here are some that we use all the time.

```
- TUNER          (contributed library)
- SOO            (contributed library)
- PROGINFO       (contributed library)
- LOOK
```

There are quite a few others available too.

We also use the OPT/APS software from HP, but we hve found it rather more comple than the above. Also the best bits are not regularly available to customers, being regarded as consulting tools needing HP's own staff to use them. (This can be expensive).

9. MPE-V (I think)

Everytime I think I've got the 3000 sorted out, along comes something new to change things.

Here are some of the major enhancements that MPE-V should bring. Firstly increased table sizes.

CST up from 192 to 2048 BUT any one user can only use 512 entries of which 256 must be HP library segments.

Maximum program size is up from 2MByte to 8Mbyte.

```
PCB up from 256 to 1024
DST up from 1024 to 4096
Disc request queue up from 256 to 900
I/O request queue up from 256 to 1300
More entries in file system director
DRT changes  -  up to a total of c. 500 devices of which
                336 can be local terminals
                50 can be DS sessions.
```

Secondly, disc cache and RPS on the CS/80 discs.

Thirdly, new communications support for DSN/X.25 and (maybe) a local area network.

Fourthly, faster file backup routines.

As a result of the above, MPE-V will also bring

- larger system code segments (all these extra facilities have to go somewhere)
- more memory utilisation (including DEGRADED performance on small memory systems)
- more places to go wrong if you're not careful.

What MPE-V will NOT bring is:

- larger code and data segments
- larger real memory address space (I might be wrong on this. If the 4 Mbyte board really is coming, it will need an enlarged memory address capability)
- downwards compatibility to MPE-IV. There will be an intermediate product (MPE-VR?) for users of 3X systems and the older Series II/III without the increased table size features and the RPS driver modification.

You will, however, be able to get more out of your 4X system without finding the space, power, A/c and money for a 6X upgrade.

11. Conclusions

I hope you now feel that the effort of a sizing and performance review is worthwhile. Can I also suggest that it is worth doing BEFORE you have a problem to solve, and that this 'prevention mode' exercise could actually head off a problem? I am sure that installations that know what is going on with their system fare better when bidding for new resources than those that do not. They probably give a better service to application's users as well. This need not cost a lot of money and nearly always yields quick benefits. So think about:

- Workload and Workmix
- Transaction volume
- Good Application Design
- Performance Monitoring tools

and you will be able to SEE where all your hardware went, perhaps even get some of it back to work!

John S. Parkinson graduated in the UK in 1972 with a BS in mathematics and an MS in information sciences. After 5 years working on a variety of health care related system projects for the UK National Health Service, he joined Sifo Sytems, a specialist health care systems house and consultancy, where he is now General Manager and head of Product Development. His experience with the HP 3000 started in 1978, when SIFO became an OEM and since then he has been responsible for the installation of systems in the UK, Europe and the Middle East. He has presented papers at several UK and European User Group Meetings, and runs a Sizing and Performance Seminar for users in the UK.

QUALITY DOCUMENTATION -- A HOW-TO

by
Robyn
McIntyre

Good documentation begins with a plan. While the plan presented in this paper isn't by any means definitive, it should help you get started and keep you going while you develop a formula of your own. It consists of four main elements:

- Defining the need
- Planning the document
- Writing the document
- Maintaining the document

DEFINING THE NEEDS

Before anything else, you should define the needs the document will be designed to meet. Doing that involves asking three questions:

- What is being documented?
- Who will be reading it?
- Will it also serve as a sales tool?

What is being documented?

What purpose will the document serve? Will it be used as a training manual, a procedures manual, a reference manual?

Who will be reading it?

Will the document be written to serve a technical audience, a non-technical audience, or someone in-between?

Will it also serve as a sales tool?

Almost every document meant for commercial purposes has the capacity to be used as a sales tool. Well-written documentation is an increasingly important factor in the buying and selling of software and hardware products. But some documents can be more sales-oriented than others. A manager's guide, for instance, might be expected to contain more information about the features, benefits, and reliability of a system than a key-entry operator's manual.

When you have defined the needs the document will serve, you will be better able to focus on what it must contain. And that will be a guide to you as you go on to the next stage of creating a document: Planning.

PLANNING

Planning is probably the most important task in the process of developing a document. For many it's also the most time-consuming and boring of the tasks (having a big wedding is often more fun than planning it). But without good groundwork, a document (like a wedding) can turn out to be a disaster. To plan for a good document involves four steps:

- Formatting
- Research
- Writing an Outline
- Estimating Writing Time

Formatting

If your company provides you with a word-processing system, but doesn't provide you with a "style" guide, formatting will be

Planning
FORMATTING

your first step in planning the document: You need to know where headers will appear, what your margins will be. Rough drafts needn't be that rough, and even with a wp system, you can save time by not having to go back and change margins and such. If you're starting from scratch, designing the "look" of your document may involve looking at other manuals, and consulting publications on technical writing. The main idea is to produce a document that presents its material in a way that is both effective and pleasing to the eye. The look is important because if it doesn't appear inviting, no one is willingly going to read it. While you're designing your format, take into consideration the type of media that will be used. For the purposes of this discussion, I'm assuming the final result will be a manual. But obviously your approach will be different if the final "document" will be microfilm, video tape, or plastic-coated cards like the instructions found on copy machines.

Whatever media you use, there are some elements of formatting that provide a "cleaner" appearance and better communication:

- headers
- call-outs
- bullets
- narrow columns
- simple pagination
- visual transitions
- illustrations

headers

Almost everyone will make use of one kind of header or another, but many will not consider the real value of a header as a signpost. A bolded header on every page will help the reader to determine quickly what information is to be found there. Without having to dig into the text. If possible, place a subhead under the page header (see "FORMATTING" at the top of this page) to further define the information being discussed.

call-outs

Call-outs, like the ones found in the margins of this paper, will help your reader zero-in on the specific information he wants.

bullets

Imagine a string of grocery item names in paragraph form... The moment you see it, you begin to mentally separate the items into a list. Don't make your readers do this extra work. List-type information is easier to read and to understand if it is shown in list form, and bullets will give the information continuity and emphasis.

narrow columns

Paragraphs should be short and fairly narrow. Text is easier to read when the information is broken up into small pieces. Without anything to divide it up, the text becomes confusing, the eye tires, and the reader starts to wonder whether it is worth the effort.

simple pagination

The simplest page numbering system is the best. Finding page 1-10 rather than A-202.003 is not only easier on the reader when he's looking for something, but will make maintaining the document an easier task for you.

visual transitions

Visual transitions are another way of breaking the text up into more easily-digestible pieces. They may be borders, or horizontally-ruled lines.

Planning
FORMATTING

illustrations

If your budget or technical resources permit, include as many illustrations as you can. They will make your point faster, and if done well, will aid your reader in more thoroughly comprehending the material. In addition, they make the document more attractive, and serve as visual transitions.

To clarify the points I've made about formatting, let's look at a couple of illustrations.

Figure 1.

THE AGING REPORT

The Aging report lists all unpaid invoices and how long they have been unpaid. If an unpaid invoice has a discount, the discounted amount appears in this report. It can show either: 1)all unpaid invoices, or 2) all unpaid invoices with a "marked for payment" date the same as, or older than the aging date. The aging date automatically defaults to the current date, but can be changed to any other. Additions or deletions can be made to the report by using the "Mark invoices for payment" operation, changing the "date to pay" field.

Enter the aging date. The default is today's date, but you can change it to any other.

If you don't want invoice details included in the report, enter N. If you do want them included, press TAB to skip this field and accept the default (Y).

If you want only invoices to be paid on or before the current aging date, enter Y. If you want all invoices included, press TAB to skip this field and accept the default (N).

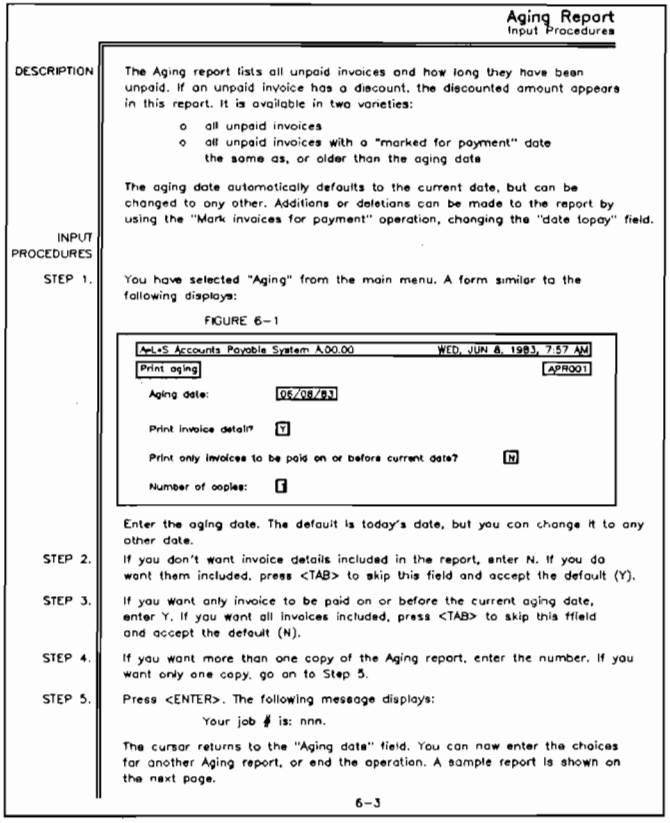
If you want more than one copy of the Aging report, enter the number. If you want only one copy, skip this field.

Press ENTER. The screen shows: Your job# is: nnn. The cursor returns to the Aging date field. You can now enter the choices for another Aging report. Or end the operation.

Compare Figure 1 to Figure 2, an example of better formatting, and the inadequacy of Figure 1's presentation is thrown into sharper focus.

Planning
FORMATTING

Figure 2.



Aging Report
Input Procedures

DESCRIPTION

The Aging report lists all unpaid invoices and how long they have been unpaid. If an unpaid invoice has a discount, the discounted amount appears in this report. It is available in two varieties:

- o all unpaid invoices
- o all unpaid invoices with a "marked for payment" date the same as, or older than the aging date

The aging date automatically defaults to the current date, but can be changed to any other. Additions or deletions can be made to the report by using the "Mark invoices for payment" operation, changing the "date topay" field.

INPUT PROCEDURES

STEP 1. You have selected "Aging" from the main menu. A form similar to the following displays:

FIGURE 6-1

Enter the aging date. The default is today's date, but you can change it to any other date.

STEP 2. If you don't want invoice details included in the report, enter N. If you do want them included, press <TAB> to skip this field and accept the default (Y).

STEP 3. If you want only invoice to be paid on or before the current aging date, enter Y. If you want all invoices included, press <TAB> to skip this ffield and accept the default (N).

STEP 4. If you want more than one copy of the Aging report, enter the number. If you want only one copy, go on to Step 5.

STEP 5. Press <ENTER>. The following message displays:

Your job # is: nnn.

The cursor returns to the "Aging date" field. You can now enter the choices for another Aging report, or end the operation. A sample report is shown on the next page.

6-3

Though Figure 2 doesn't have bolded headings (it was created using HPDRAW), it still entices the eye easily through the information it presents by the use of call-outs, bulleting, and obvious paragraphs. A likeness of a screen the user might see is included, and an allusion is made to a report sample. Whenever possible, illustrations of what the reader may see on a CRT screen or in a report should be included.

Once you have at least a rough idea of what the finished document will look like (bearing in mind that minor alterations may still be necessary), you can get on to gathering the information to be presented.

Planning
Research

Research

Researching the information you'll be including in the document involves three tasks:

- defining the information to be included
- learning how the system works
- setting up interviews

defining the information to be included

If the system already exists, you'll need to gather any previous documentation written for it. Although documenting after-the-fact goes more quickly than creating documentation for a system still in the design phase, the inconsistencies you are likely to find cannot be readily corrected because the system is in use. If the system is still in the planning phase, make sure that you are included in the creation cycle; that you get a copy of any outlines, flowcharts, et cetera, which illustrate what the system will consist of. Get a copy of the bug list and ask to be kept informed about changes, fixes, and scheduled completion dates (very important, if you're accurately to estimate writing time). If regular meetings between the project team; programmers, testers, documentors, engineers, or others are held, attend them. As boring as a meeting often is, unless it's a disguise for a gripe session, some valuable information about the way the system works will be discussed there.

learning how the system really works

Approach the system from a user's perspective: sit down in front of a terminal and use it. In an ideal evironment, the completed system or the module you're working with has already been tested and released for use. To do a thorough job of documenting, you will still need to learn the system, but if a Test department exists, find out who is testing your software. The tester can often provide more detailed information about how the system works than the programmers can (especially if there are several of them coding the project). Unfortunately, of course, some companies don't have Test departments. In that case, it's crucial that you learn the system. You'll probably find several inconsistencies in menus and form names, and a few bugs that can be corrected at least in the next release. Needless to say, take lots of notes. Indicate where the way the system actually works differs from the way it was expected to work. In particular, try entering erroneous information. You may find an error message doesn't exist where it should, or (as I've seen happen) you may even blow up the system. Let me emphasize that the object of this exercise is not to catch the programmers out, but to help them make sure that the system works the way it's supposed to, with no unpleasant surprises. And don't hesitate to indicate changes you feel might make using the system easier; fewer menus, more menus, changing the name of an operation, and so forth. Remember that you are representing the user, and you should keep their best interests at heart.

setting up interviews

Make a list of the people you should talk to before and during the document's creation. This includes the programmers and testers, and if the system is in use (and you have access to them), the people who use it. Talking to the users is especially important because, after the documentation is printed and released, you will need feedback on its accuracy. We'll discuss that further in Maintaining the Document. For now, make a list of people to talk to, and when you have several questions that need answers, schedule interviews with the appropriate persons.

Planning
Writing an Outline

Writing an outline

Now that you've determined what the document will look like and what it will contain, it's time to put together an outline.

Begin by determining the organization of the material: although it may sometimes be advantageous to begin at the main menu and go right down the list, it often makes more sense to find out which operations or modules will probably be used most frequently and therefore should be more prominently featured. For example, if reports aren't scattered hither and yon, it might make sense to group the procedures for printing them in a chapter of their own titled "Reports". Try to have the information flow from "most-used" to "least-used". If the document will be a tutorial, begin with the simple and progress gracefully to the complex. For example, help the user to be successful at creating and printing a basic letter before you discuss intricate formatting commands and techniques for mass-mailings.

Figure 3 is an example of an outline for an engineering system user's guide.

Figure 3.

EDS USER'S GUIDE -- OUTLINE

Chapter 1. Introduction

A. Description
  1. What EDS is and does
  2. What information is stored using EDS
     a. the Part Catalog
     b. the Parts List
     c. Drawing Information
     d. Engineering Change Requests
B. The operator's role -- how information is stored and retrieved
C. Documentation conventions

Chapter 2. Working With Computers

A. The CRT
  1. General description
  2. How to use it
B. Menus
C. Screen names

Chapter 3. Logging On and Other Basics

A. Logging on
B. Selecting an operation
C. Ending an operation

Chapter 4. The Part Catalog

A. Description
  1. What the Part Catalog is and does
  2. What information is stored here
B. Operations Catalog
  1. A general description of all Part Catalog operations
C. Input Procedures
  1. Add/change part information
     a. input procedures
     b. error messages

Looking at Figure 3, you'll notice that Chapters 2 and 3 consist of basic information needed to use the system. The step-by-step instructions for performing operations begin in Chapter 4, and once the key entry clerk has mastered the basics, Chapters 2 and 3 can be skipped over with no loss of information. Chapter 1, of course, is an introduction to the system. In this, the user's manual, the introduction is less detailed than might be designed for a manager's guide, since the key entry clerk needs less background and interpretive information to do his job than someone who must plan for an entire department.

Planning
Writing an Outline

Your own outline may be more or less detailed, depending on how much information you feel is necessary to remind you about what must be included and where. Try to keep wide left and right margins because you will probably scrawl notes as you progress. And if you use a word processing system to create the outline, leave it on the system. Information may arrive late and make revisions necessary.

When you're satisfied with the outline, have it approved. When I worked for AM Jacquard, approval of an outline was required from the manager of documentation. Yours may be approved by the DP manager, programmer, or no one. The important thing is to have someone familiar with the material review the outline and give you some feedback about how the information will be presented: Did you leave something out? Does the material progress logically? Approval of the outline can be especially important if you are working closely with the programmer and user to design a new system. Your outline may be the first opportunity they have to see, in a rough way, how the software will work. After reviewing it, they may decide some operations have been left out, and others are redundant. Using it, they can decide whether the project is on-target. And you can fine-tune your organization of the information.

Now that you have pretty much decided what the document will look like, begun your research, and have gathered enough information to decide how the document will be organized, it's time to estimate how long it will take to write it.

Estimating Writing Time

If you are documenting software that already exists, and has been released to you for documentation, estimating the time needed to write it will be fairly straightforward. Figure 4 shows a formula you can use (I'm indebted to Progressive Communications of Colorado for developing this specific formula).

Figure 4.

TASK	TIME ESTIMATE
First Draft	Research time + writing time + 15%
Second Draft	25% of time to create 1st draft
Total writing time	1st draft time + 2nd draft time
Total proofing/ editing time	25% of total writing time:
1st draft	75% of total proofing/editing time
2nd draft	25% of total proofing/editing time
Total typing time	35% of total writing time (automated typing is assumed)
1st draft	75% of total typing time
2nd draft	25% of total typing time
Review	Double the reviewer's estimate!

Planning
Estimating Writing Time

Now let's take a look at what this formula looks like when translated into reality. Figure 5 shows the time estimated for completing a first and second draft for a report guide containing information about fifty reports.

Figure 5.

Writing/Production Time
Report Guide – 50 reports

TASK	HOURS
First draft = 1 hr. research + 2 hrs. writing + 0.45 hrs. (extra 15%) =	3.45
Second draft = 25% of 3.45 hrs. =	0.86
Total writing time = first draft (3.5 hrs.) + second draft (0.86 hrs.) =	4.3
Total proofing/editing = 25% of 4.3 hrs. = (1st draft proof/edit = 0.8 hrs.) (2nd draft proof/edit = 0.27 hrs.)	1.07
Total typing time =35% of 4.3 hrs. = (1st draft typing = 1.13 hrs.) (2nd draft typing = 0.38 hrs.)	1.5
Total writing/production time to finished 2nd draft (one report) =	6.9 hrs. per report
Total writing/production time for 50 reports = 6.9 hrs. x 50 (reports) =	344 hours (div. by 40) = 8.6 weeks

As you can see, using this formula, the estimated time for completing the work is over eight and one-half weeks at five days a week.

When estimating time for the first draft, take into consideration the time you'll need to arrange for, or construct roughs of tables, report samples, and examples. Will they be hand-drawn or printer-generated and reconstructed by the art department? Must you include screens in the draft or can you just indicate where they will be included later? If you are your company's art department, reports, tables, examples, and screens will affect how long it will take to complete the final draft. So before you estimate when writing will be completed, make a list of all of the illustrations you'll need and what it will take to get them all together. If other people are involved in the process, take into account that even though you are all working on the same material, you won't work at the same speed.

As handy as the formula is, though, if you are documenting software still in development, you may not be able to use the

Writing

Plan to Write Poorly and Edit Well

formula to estimate a finish time for a completed document. The program may not be ready when you are. In cases like this, it's best to work in a module form, breaking down the time necessary for documenting the entire program into its component parts. For example, if I'm writing an instruction manual for an engineering system and I know the reports are finished, though nothing else is, I can give an estimated time for documenting the report section. Documenting software as it's released means you and the programmer will have to work closely; you're depending on him to stick to his schedule so that you can stick to yours. If you have a situation like this, make as sure as you can that you have a means of prodding him if he slacks off. And that the prodding is done through a third person such as the project leader, DP manager, or software manager. This will ensure that the programmer won't duck down a stairway when he sees you in the hall.

I'd like to note at this point that the formula for estimating writing time may become entirely useless to you if you must work backward from a due date. But (at least in my experience) unless the due date is entirely unrealistic, you'll probably have more time than you need to finish up.

Finally, after completing the four steps necessary to planning the document (formatting, research, writing the outline, and estimating writing time), you can get down to writing.

WRITING

Although planning is the most time-consuming and possibly boring task connected with creating a document, most people (including me) will do almost anything to avoid actually beginning to write. I sharpen pencils (seldom needed when using TDP), get a cup of coffee, ask my friend Cathleen about her wedding plans; anything to avoid that moment when I face the terminal and begin the first sentence. I don't know why this is so; I've been writing user manuals for six years, and fiction for longer than that, but I always have this reaction to a new piece of work. So if you have that same feeling, don't think you're alone. Writing is hard work. You are creating something. Maybe it isn't destined to win you the Pulitzer, but it will still have your own stamp on it, and it will take energy (a lot) to produce.

Plan to write poorly

To minimize start-up anxiety, keep a couple of things in mind: Since you're writing from an outline, you can begin wherever you feel most comfortable. Also, you can plan to write poorly. This means, don't be too concerned with the style and sentence structure in the first draft. If you have stockpiled material from various sources, throw it in without a thought for how it sounds and whether the transitions are rough. You can always (should always) go back and clean up the text. The main thing is to get started and get it all out. Still, planning to write poorly doesn't mean you shouldn't do the best you can while you're at it; the better your organization and writing is to start with, the less editing it will require. And if you keep the following elements in mind as you write the draft, you'll be ahead of the game:

- tell him, tell him, tell him
- control jargon
- use stock phrases
- keep it conversational

Writing

Plan to Write Poorly and Edit Well

- stay active
- keep it simple
- build signposts
- avoid cuteness
- include necessary references
- provide thorough examples

tell him, tell him, tell him

This refers to an old saying in military teaching: Tell them what you're going to tell them, tell them, tell them what you've told them. Translated, it means you'll start out by telling the reader what he can expect to find in a chapter, then you'll tell him what you said you would, then you'll tell him what it is he just learned. This redundancy will make it easier for the reader to remember what he's been told, and it will help you move logically from one place to the next. How about a simple example:

Figure 6.

OPERATING A LIGHT SWITCH

Description — This section describes the steps necessary for turning an and off room lights using a light switch. It is presented in these sections:

- o Turning on a switch
- o Turning off a switch
- o Error Conditions

Turning on a switch:

Step 1. Locate light switch (see illustration on page A-23).

Step 2. Place forefinger of either hand under switch.

Step 3. Using a brushing motion, and keeping forefinger firmly against switch, lift up. Light should be on.

Now that you have learned to turn on a light using a light switch, you can go on to learning to turn it off.

Turning off a switch:

Step 1. Locate light switch (see illustration on page A-23).

You may think that repeating yourself is likely to irritate the reader, but it doesn't work out that way. It will actually make it easier for the reader to remember what you've told him. Just remember while you're repeating yourself to avoid talking gibberish.

control jargon

For many readers, jargon is gibberish. They don't have your job or associates, so they probably don't hear the same catch phrases every day. If you insist on using buzzwords to describe processes, they will probably give up in disgust and frustration. Sometimes (as with data processing), some jargon is unavoidable. A new data entry clerk will more than likely have to learn to use the word "CRT" even if he doesn't know its exact definition. When you must use jargon, explain it. Or include a glossary of terms

Writing

Plan to Write Poorly and Edit Well

used frequently in the documentation. Try to keep the number of terms small, though; people visiting foreign shores don't usually immerse themselves in the language of the country they're visiting. One thing that will help are stock phrases.

stock phrases

Stock phrases such as, "The operations performed in the ------ feature are:", or "The cursor returns to the ------- field" become familiar to the reader and make him more comfortable with the instructions. They also make writing the document go faster because you aren't creating a new phrase for each function to be performed. Don't worry about creating these stock phrases before you begin your first draft. If you look for them as you write, they will more or less identify themselves to you, and indicate where they belong. But new text or stock phrase, keep it conversational.

keep it conversational

A document explaining a technical process is bound to be dry; don't completely dehydrate it by using a tone more suited to addressing a gathering of stuffed shirts. Keep it friendly, keep it light, keep it conversational. You're explaining something to a friend, face to face. Keep your text on that level.

stay active

One way to keep the tone conversational and interesting is to stay in the active tense. My english handbook defines active voice in writing as "making the subject of a sentence the doer of the action", as in

"Ned washed the car."

When the subject of the sentence is the receiver of the action, then you have passive voice, as in

"The car was washed by Ned."

Just from these two examples it's obvious that the first sentence (active voice) not only uses less words, but is more lively and interesting. If you have been taught to use passive voice, you may find it difficult to catch yourself at it. I recommend that you do two things: Get a reference book on english usage and study up on active and passive voice, and find someone who can spot passive voice when she sees it. Ask her to read through your draft when it's finished and indicate the passive constructions. After some help you'll find it easier to recognize them on your own, and then you'll find them cropping up a lot less often.

keep it simple

Passive construction is one of the chief reasons that many scientific journals and publish-or-perish papers are so darned boring. Another reason is that, for one reason or another, the authors seem to feel their viewpoints won't be taken seriously unless they throw in a lot of jaw-breaking vocabulary. Maybe so, but your reading audience is more likely to be interested in learning how something works, and not in how educated you are. This doesn't mean that you can't use polysyllabics, but it does mean that you must take your reader into account. Go back to where you defined the needs the document is being designed to meet. Who is going to read it? If it's designed for managers, you could use a more complicated vocabulary. But manager or key entry clerk, the issue isn't intelligence but time. Will the reader have the time and patience to deal with a lot of ten-dollar words. The

Writing

Plan to Write Poorly and Edit Well

answer is probably "No". So keep it simple. Don't patronize the reader, but remember that he has to get through an awful lot of technical information in as short a time as possible. Make it easy for him to cut through the text by saying what you've got to say in the plainest possible way.

build signposts

And while you're building plain text, you might be constructing a few signposts. Signposts are transitions; the sentences that lead the reader from one section to another. Try to make the transition an easy one; an abrupt transition will shock and confuse a reader enough to make him lose his concentration. Once concentration is lost, it's difficult to regain. Reading, as well as writing, is an energy-consuming task. Many people wouldn't read at all if they didn't have to, and if you make it tough for them, they certainly won't read your work. Which brings us to another concentration-breaker: cuteness.

avoid cuteness

Cute drawings and sayings can be useful in some circumstances; sales material, mostly. But cute drawings have three failings:

- they break the reader's concentration
- after a few readings, they stop being cute
- the style helps to "date" the material

On the whole, your document will be better off without drawings about "Willie Workorder" or "Penny Purchaser". It will also be better off if you use the space to include necessary references.

include necessary reference

What is a necessary reference? A necessary reference is a bit of information about another command or procedure or piece of hardware that has a bearing on the information you're currently describing. A good example is something that recently happened at my office: The department secretary was formatting a document using TDP. The end of the document contained a command to include another TDP file. It wouldn't format. Each time she tried, the formatting would abort, and it took her, me, and the DP manager 15 minutes to figure out that it was aborting because the file to be included specified an environment file in the first line, confusing TDP. Using hindsight, the problem might seem obvious. But first you have to make the connection. How much simpler to avoid the problem entirely by including a short note in the documentation for the IN command stating that the file to be included cannot contain an environment file specification! Don't make things any harder for the reader than you must. If some command or procedure affects another command or procedure, indicate it.

provide thorough examples

Another way you can make things easier for the reader, is to provide thorough examples. It's not necessary to try to cover every contingency, but the examples you include should provide enough information to enable the reader to use the command or procedure with confidence. Don't give weak or incomplete examples. Incomplete information causes readers to give up, and a command or procedure that isn't used isn't really useful.

Okay, the first draft is finished. You've planned to write poorly, but saved yourself a lot of grief by remembering to:

Writing

Plan to Write Poorly and Edit Well

- tell him, tell him, tell him
- control jargon
- use stock phrases
- keep it conversational
- stay active
- keep it simple
- build signposts
- avoid cuteness
- include necessary references
- and provide thorough examples

Now what? Now, of course, you edit.

Edit well

By editing, I don't mean proofreading. If possible, someone else should check to make sure the commas are in the right places. No, by editing, I mean reading through the work to see if it makes logical sense; does one thing follow another in a sensible manner so that the reader is led gently through the material to its obvious conclusion? From first entering the data to printing the report? From writing a basic letter to creating complex contracts from boilerplate?

To save your sanity, you will probably want to break down the editing process into several readings:

- check for organization
- check for transitions
- check for sentence construction
- check for completeness and content

Unless you used to teach grammar, you could probably use a good reference book. You'll find plenty of them in the reference book section of your local bookstore, but if you can, I recommend you buy Strunk and White's Elements of Style (MacMillan Company, publishers). A good reference book will become invaluable. After a few documents, you'll develop a feel for sentence variety, agreements, and other fun grammatical stuff, and you'll need your reference book less often.

Proofreading

Once the editing has been done on the final draft, the document can go to the art department, or official proofreader, or whomever can do a good job at ferreting out misspellings, punctuation errors, and the like. Try not to do this job yourself, if you can avoid it. After being totally involved with the document, you can't see it the way someone else would. If you must proof it yourself, try to give it some cooling off time. Drop it in a drawer for a couple of weeks. When you do see it again, it will seem different to you.

Miscellaneous

Before we leave editing for maintenance, I'd like to make some miscellaneous points about reviewing documentation:

Preliminary Stamp

Try to have a stamp made up to say something like, "Preliminary Version" or "Rough Draft". Use them when you release the first draft of a document for review. Sometimes rough drafts get copied or lost, and if you're doing in-house documentation, you won't want anyone using the rough draft and thinking it's gospel. Also, stamping it "Preliminary Draft" or something similar will remind reviewers that they are not looking at the final version.

Maintenance

You might not think that would be necessary, but you would be surprised at the number of times I've had to tell someone, "Don't bother about that; the work isn't finished."

Reviewers

Speaking of reviewers, make sure a technical reviewer knows what his job is: reviewing a document for technical accuracy. Many reviewers assume that they should help you out by editing your work and proofing it, too. This can be maddening, since you've already budgeted time for proofreading and editing. And if the reviewer wastes time on these tasks it will only be longer before you can get to work on the technical content revisions. That's assuming that the reviewer knows his editing and proofing stuff. If you've explained this to the reviewer, and gotten nowhere, try Progressive Communication's suggestion: have him indicate technical content changes in one color and editing/proofing changes in another color. That way, you can easily identify the editing/proofing marks, and ignore them.

Indexing

One thing you shouldn't ignore is the Index. Any complicated piece of documentation should have an index, and as complete an index as possible. What constitutes a complicated work? Anything written where there is so much information valuable to the reader that you can't list it all in the table of contents, or where an item is discussed in more than one chapter or section.

When in doubt about whether or not to include an item in the index, I recommend including it. It's easy to skip information you don't want, but very irritating to be unable to find what you do want.

MAINTENANCE

Finally, you've dealt with the reviewers and gotten the document formatted, printed, and released. Now you must maintain it.

Bug Lists

If you're part of a large programming/documentation staff - as in a research and development shop - word will come down to you via the bug/enhancements list about what changes you can expect the document to need for a specific release.

If you are a one-person documentation department, there isn't any reason why you shouldn't ask for a bug/enhancement list if one doesn't already exist.

However your company works, you must be informed about expected changes well in advance or you can't plan for writing them up. Don't let your manager get away with coming up and saying, "Oh, by the way, we changed the Blank Report six months ago and forgot to tell you. I need a revised copy of the Report Guide for tomorrow's staff meeting. Get to it, will you?" You'll only get ulcers. I may be predjudiced, but if somebody hires me for a job, I expect them to let me do it. And do it right. To that end, I also expect some management cooperation. So if you're currently working for a company without a method for informing you about programming changes that will affect documentation, come up with one, or work with your boss and the programmers to come up with one, and insist that they use it.

Feedback

Another type of documentation change to be considered is the change indicated by user feedback. Using postage-paid forms, telephone or in-person interviews, it can be possible to find out

Maintenance

what the reader thinks of the way you organized and presented the information in the document. This kind of data is often difficult to get, though, since many readers will find work-arounds for inaccurate documentation without telling the vendor that they're necessary.

Modular Approach

Once you've become informed about expected changes, you've got to make plans for ensuring the changes make it into the documentation. If you're working for a software/hardware vendor, the changes should be included in the appropriate release. And if you've used a modular approach to documentation (Report Guide, Input Procedures, etc.), you should just be able to instruct the reader to remove Section X. from the manual and replace it with the new Section X. This cuts down costs, and makes both your task and the reader's a lot easier.

In-House Library

In-house documentation should be catalogued in a company library. The documentation department, or document control department, or whoever, should have a list of all manuals produced and where in the company they're located. This will speed the process of replacing the old with the new.

Page Breaks

If, when you wrote the document, you avoided putting in hard-coded page breaks except where necessary, the task of adding or subtracting text will be eased. (Of course, by hard-coded page breaks I mean entering "\NEW" commands or anything else that would force a new page.)

Table of Contents

While you're listing the parts of the document to be added to, or subtracted from, don't leave the table of contents out. Changes in the document may mean additions or deletions here, too.

Index

And by all means, don't forget about the index. If you're adding new information, the index may also require additions or deletions in even more detail than the table of contents.

CONCLUSION

Well, we've finally reached the end of this paper, and I still feel I could have been more specific. Unfortunately, there is not time or room for an in-depth study of the art of writing quality documentation. That would be (and has been) a subject for an entire manual, three-day seminar, and even a college quarter. I hope, though, that I have given you a good basis to start from by stating that the formula for writing quality documentation consists of four elements:

- Defining the need
- Planning the document
- Writing the document
- Maintaining the document

As I have said, while planning the document is the more time-consuming task, writing the document is the most difficult. So go easy on yourself if you're new to tech writing; realize that it takes time to learn, and lots of practice to do something well. Don't be afraid to borrow ideas (unless they're copyrighted); each writer has his or her own style and will inevitably create a different product using the same tools. If you've been a technical writer for awhile, and you enjoy it, you've already noted the ideas most interesting to you and no doubt made plans to incorporate

Conclusion

them into your next work. But beginner or pro, if anything I've covered in this paper helps you make your documentation clearer to the reader, I'm satisfied. Because that was always my main objective.

Robyn McIntyre is Senior Technical Writer for Infotek Systems in Anaheim. When not staring blankly at CRT screens or waylaying programmers, she can often be found researching her current project, "The Best Chocolate in L.A. (County)".

DISASTER RECOVERY - PLANNING FOR THE UNPLANNED!

A major data processing disaster is something most operations organizations never experience. However, there are various degress of severity for disasters ranging from complete loss of facilities, equipment and records, to temporary malfunction of time-critical equipment. The author proposes to first discuss the types and levels of potential disasters siting various case study examples (including one which happened in his own facility). Then, the author will identify the key elements necessary to develop a Disaster Recovery Plan, emphasizing how to prepare a DR Plan that is realistic. Last, the author will discuss approaches for periodically testing the DR Plan.

Disaster Recovery - Planning for the Unplanned!

Richard A. Savaiano
Principal
Industrial Management Associates
Manhattan Beach, California

INTRODUCTION

Although a major data processing catastrophe is something most organizations never experience, there are various degrees of severity for computer center disasters. These range from temporary malfunctions of time-critical equipment to complete loss of facilities, equipment and records. Disasters such as fire, flood, environmental problems, hardware and software problems, and sabotage can and do happen. Management often tends to ignore these possibilities. If management reviews the impact of data processing operations disasters, and consciously chooses not to plan for such major disasters, then perhaps the risk is not serious enough to warrant taking precautions. However, with today's corporate dependancy upon computer centers and information processing, the more typical situation indicates that when the computer center goes down, all company work is affected.

PURPOSE OF PRESENTATION

The purpose of this presentation is threefold:

1. To emphasize the need for and the importance of Disaster Recovery Planning.
2. To identify the key elements of a Disaster Recovery Plan.
3. To describe an approach for insuring that the organization's Disaster Recovery Plan will work.

In today's competitive, fast-moving, constantly changing business environment, the importance of computers and information systems is quite evident. However, it is estimated that less than 50% of the Fortune 1000 companies have a plan in place to insure continued availability of these valuable corporate resources. In addition, it is estimated that only one-half of the disaster recovery plans in place are workable. For smaller companies the percentages indicate an even worse situation.

POTENTIAL TYPES OF DISASTERS

Disasters occur infrequently, thus management may tend to deemphasize the immediate need for a recovery plan. The potential causes of disasters range from unique

and unusual weather conditions to deliberate employee sabotage. The list of typical causes include fire, storms, earthquakes, structural collapse, power failures, water damage, massive equipment failure, excessive heat and sabotage. The following three cases emphasize the need for Disaster Recovery Planning.

Company A:

A 300-watt light bulb was left burning over night in a supposedly fireproof tape vault for a large government computer center. The ceiling material next to the light began to smolder. The next morning when a computer operator opened the vault door to retrieve required tapes, the oxygen from outside the vault caused the fire to ignite in seconds. Attempts to turn off power to the computer center failed because the only power shutoff switch was located in a remote area behind the fire. Thus, fire personnel were pouring water on electrical equipment that was still running. Before the fire was over, the entire computer center was destroyed.

Three major problems contributed to this disaster. First, the material used in the vault should have been fireproof, not merely fire resistant. Second, the master electrical shutoff switch should have been located near

an outside door. Last, the computer center should have been designed to allow easier access by fire personnel. Proper Disaster Recovery Planning would have identified these problems.

Company B:

In the early morning hours a fire started in the warehouse of a medium-sized manufacturing company. Before fire personnel arrived, the blaze spread to the administration building of the company. Upon review of the damage, the president of the company quickly realized that most of the company's accounting records, as well as all of the data processing system, had been destroyed. The loss put the company out of business.

The major problem at this facility was the lack of a Disaster Recovery Plan and the associated remote site storage of critical and necessary information. With the proper backup facilities and procedures in place, the company could have survived.

Company C:

A machine caught fire in the production area of a small manufacturing plant. Because of the close proximity to highly combustible material, fire personnel reacted

promptly and efficiently dispensing thousands of gallons of water into the manufacturing facility. Water also flooded the adjacent administrative offices and the computer center. Fortunately, an alert computer operator followed the published emergency shutdown procedures eliminating the potential for fire and electrical damage in the computer room.

The next day, computer service personnel inspected all computer equipment, giving the go ahead to start up the facility only eight working hours after the fire. Published and practiced disaster procedures prevented a potential crisis.

Examples of disasters and potential disasters, such as the ones discussed here, re-emphasize the need to establish a Disaster Recovery Plan. An IBM study shows that over a ten year period from 1968 to 1978 there were more than 350 major data processing catastrophes.

PHASES AND ELEMENTS OF A DRP

The most important element of any Disaster Recovery Plan is the identification of the critical company functions. The primary objective after a disaster should be to restore

these critical functions. To accomplish these objectives, the designer of a Disaster Recovery Plan must understand two things: the phases of events in Disaster Recovery and the key elements contained in a Disaster Recovery Plan.

The Five Phases in Disaster Recovery are:

1. Preparation
2. Protection
3. Recovery
4. Critical Operations
5. Normal Operations

The Key Elements Contained in a Disaster Recovery Plan Include:

1. Introduction and Assumptions
2. Staffing
3. Hardware
4. System Software
5. Application Software
6. Data Files
7. Facilities
8. Operational Procedures
9. Transportation
10. Supplies

TESTING YOUR DISASTER RECOVERY PLAN

A major downfall of many Disaster Recovery Plans is the lack of a periodic review and testing of the procedures. A disaster should be simulated so that the recovery procedures can be tested, insuring that backup facilities work in the event of a crisis. Successful operation at an alternate computer site, using remote site backup equipment, programs, files and documentation is insurance to minimize the impact of a real disaster. Specifically, periodic testing verifies the availability of an alternate site and insures the compatibility of software. In addition, data records, supplies and operational procedures are checked.

SUMMARY

Disaster Recovery Planning is a function that many organizations will never use. However, for those who choose not to spend the time and effort developing and maintaining a DRP, the potential risk is high.

As evidenced by the examples of computer center disasters, the time of an emergency is not the time to develop an emergency plan. The likelihood of a serious computer disaster in any one company during a year is slight. However, with the increasing dependancy of businesses upon computer centers, distributed processing, executive workstations and timely information processing, the need for a recovery plan is great. Once a disaster strikes, your alternatives are limited. As a data processing manager, now is the time to PLAN FOR THE UNPLANNED!

SELECTED REFERENCES

1. Friedman, Stanley. "Just in Case ... Planning for a Disaster," Small Systems World. April, 1983, pages 28 - 31.

2. Perry, William E. Computer Control and Security. New York, New York: John Wiley and Sons, Inc., 1981.

3. Porter, W. Thomas and William Perry. EDP Controls and Auditing. Boston, Massachusetts: Kent Publishing Company, 1981.

4. Tarrington, Gary and Walter Ulrich. "Insuring the Unthinkable," Computerworld. August 24, 1983, pages 48 - 51.

5. Van Tassel, Dennis. Computer Security Management. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1972.

BIOGRAPHY

Richard A. Savaiano is a principal with Industrial Management Associates, a management consulting and systems development firm located in Manhattan Beach, California. Specializing in the design, development and implementation of management solutions, Mr. Savaiano provides consulting services to a wide range of clients. In addition, Dick is a frequent speaker at management conferences, educational seminars and association meetings. He can be reached at (213) 545-3929.

The Three Bears of IMAGE

Fred White
Adager

INTRODUCTION

Software designers, whatever the product, hopefully provide a variety of features which they believe are important to user acceptance of the product.

In many cases, the implementation of a feature is optimized for the use envisioned by the implementers. Conversely, the implementation may be sub-optimized for use other than as intended.

Traditionally, product manuals seldom (if ever) include motivational discussions of product features so that users are not warned about sub-optimal uses of the product features.

In some cases the sub-optimal use of features may have no noticeable effect on throughput or response time. In others the effect may be disastrous.

Two features of IMAGE/3000 whose sub-optimal use can be disastrous are "integer keys" and "sorted paths". For the purposes of this paper, these two represent, respectively, PAPA BEAR and MAMA BEAR. Each is a very deep pitfall and extricating yourself from either can be very expensive.

BABY BEAR is represented by "paths", another feature whose misuse, while normally not disastrous, may have a negative effect on response time and/or throughput. A discussion of the use of paths is included to justify the title and because it should be of general interest.

BACKGROUND

"Detail" datasets were intended as repositories for records having generally no unique identifying characteristic (field value) and for which the primary access method would be sequential.

Each detail dataset starts as an empty file of a size large enough to meet its capacity requirements. IMAGE keeps track of the highest record number (initially zero) assigned to any record of the dataset as a result of a DBPUT. This serves as a "high-water-mark" and is analagous to the file system's EOF (end-of-file).

Stated another way, a detail dataset is similar to an ordinary MPE file in that each new record is assigned an address calculated by adding 1 to the high-water-mark. When this is done to an MPE file, MPE adds 1 to the current EOF pointer and appends the new record.

IMAGE, however, provides for the automatic re-use of space which results whenever a record is deleted. It keeps track of the reusable space by means of a push-down stack. It maintains a pointer to the newest member of this stack and each member points to an older member deeper in the stack. DBPUT always (for detail datasets) assigns the address of the newest member of this "delete chain" to the new record being "put" unless the "delete chain" is empty, in which case DBPUT increments the high-water-mark and assigns the new value of the high-water-mark as the address of the new record.

"Master" datasets were intended as repositories for records having a unique identifying characteristic (field value) and for which the primary retrieval technique would be dependent on this unique value. The IMAGE manual refers to this as calculated access.

After much discussion it was decided that two distinct "flavors" of calculated access be provided: one over which the user would have (essentially) no control and which would calculate record addresses via a hashing algorithm whose objective was to achieve a nearly uniform distribution of addresses in the face of

random or non-random key values, and another over which the user would have (essentially) absolute control in that the low-order 31 bits of the key value would determine the desired address (modulo the capacity).

For those of you familiar with "direct access" methods, this latter capability can be viewed as a generalized "direct access" method. Generalized in the sense that addresses greater than the capacity are not considered invalid but, instead, are reduced modulo the capacity. IMAGE does this by (a) subtracting 1 from the 31 bit key value, (b) dividing the result by the capacity to obtain the positive remainder and (c) adding 1 to this remainder.

It was further decided that this "direct access" technique would be used whenever the search field was defined as an item of type I, J, K or R (all of which are of binary format) while "hashing" would be used whenever the search field was defined as an item of type U, X, Z or P (none of which are of binary format).

For all of the "direct access" type keys, IMAGE treats the low-order (right-most) 31 bits as a positive integer in calculating the record address. For this reason, these keys have been dubbed "integer" keys as a way to distinguish them from "hashed" keys.

Space allocation for master datasets is completely different from that described for detail datasets.

In effect, a master dataset starts out with the high-water-mark equal to the capacity and DBPUT never appends records. Instead, the record space starts out as entirely re-usable. No "delete chain" is maintained for master datasets. Instead, IMAGE relies on a "bit map" which is maintained at the front of each block of each dataset. For master datasets, DBPUT calculates the primary address (as described above) and, after verifying that the key value is unique, attempts to place the new record at the primary address.

This attempt will succeed if and only if this new record has no synonyms. Otherwise, DBPUT assigns a secondary address physically near (hopefully) the primary address. It finds such a hole by means of a sequential (and cyclical) search starting with the block containing the current end of its synonym chain. In a master dataset which is not too full and where existing records are not "clustered" (i.e. nearly uniformly distributed) and where the "blocking factor" is not very small, this search might require zero, or only a few, disc reads.

This technique assigns synonyms to the same block or to neighboring blocks thus minimizing I/O during DBPUTs, DBFINDs and keyed DBGETS.

Having covered the pertinent differences between detail and master datasets, let us proceed to a discussion of the path feature.

Under IMAGE, a path is a relationship between a master dataset and a detail dataset. The relationship is 1-to-N (where N varies from zero to 64535) in the sense that each master record is related to N records of the detail dataset and that each record of the detail dataset is related by this path to exactly one record of the master dataset.

The N detail records related to a common master record are referred to as a chain since IMAGE links them together with backward and forward pointers. One end is referred to as the "beginning-of-chain" and the other is referred to as the "end-of-chain". New records are added to the "end-of-chain". IMAGE maintains a chain length count and pointers to the beginning- and end-of-chain in this common master record.

The common master serves as a locator record (via a DBFIND) to the corresponding detail chain. This is analogous to using the card catalog in a library to locate all books written by a particular author.

The fact that a detail dataset can have paths to more than one master dataset is analagous to the books in a library being referenced by other card catalogs such as Title or Topic.

This, together with the fact that IMAGE permits master datasets to have paths to more than one detail and have more than one path to any detail make IMAGE (along with the AUTOMATIC master feature) a very flexible 2-level network structure data base management system.

PAPA BEAR . . . the INTEGER KEY pitfall

My first live encounter with a misuse of integer keys arose in 1978.

One Friday in 1978 I received a phone call from an insurance firm in the San Francisco Bay Area. I was told that their claims application was having serious performance problems and that, in an attempt to improve the situation, they had, on the previous Friday, performed a DBUNLOAD, changed some capacities and then started a DBLOAD which did not conclude until the early hours of Tuesday morning!

They were a $100,000,000-plus company which couldn't stand the on-line response they were getting and couldn't afford losing another Monday in another vain attempt to resolve their problems.

Investigation revealed that claims information was stored in two detail datasets with paths to a shared automatic master. The search fields for these three datasets was a double integer key whose values were all of the form YYNNNNN (shown in decimal) where YY was the two-digit repesentation of the year (beginning with 71) and where each year NNNNN took on the values 00001, 00002, etc. up to 30,000.

Although the application was built on IMAGE in late 1976, the earlier claims information (from 1971 thru 1976) was loaded to be available for current access. I do not recall the exact capacity of the master dataset but, for purposes of displaying the nature of the problem (especially the fact that it didn't surface until 1978) I will assume a capacity of 370,000.

Although the number of claims per year varied the illustration will also assume that each year had 30,000.

The first claim of 1971 was claim number 7100001 which, using a capacity of 370,000, IMAGE would assign a primary address of 70,001. This is because 7,100,001 is congruent to 70,001 modulo 370,000. The 30,000 claims of 1971 were thus assigned the successive addresses 70,001 through 100,000.

Similar calculations show that the claims for each year were stored in groups of successive addresses as follows:

Year	Claim Numbers	Assigned addresses
1971	7100001-7125000	70,001-100,000
1972	7200001-7230000	170,001-200,000
1973	7300001-7330000	270,001-300,000
1974	7400001-7430000	1-30,000
1975	7500001-7530000	100,001-130,000
1976	7600001-7630000	200,001-230,000
1977	7700001-7730000	300,001-330,000

Note that no two records had the same assigned address and thus that there were no synonyms and that all DBPUTs, DBFINDs and keyed DBGETs were very fast indeed!

Now comes 1978!!!

Unfortunately 7,800,001 is congruent to 70,001 so that the first DBPUT for 1978 creates the first synonym of the master dataset. It is, in fact, a synonym of claim 7100001. Recalling that DBPUT finds an alternate location by means of a serial search, DBPUT then searches the next 60,000 records before it finds an unused address at location 130,001! Even with a blocking factor of 50, this would require 1200 additional disc reads which would make each DBPUT up to 200 times as slow as those of previous years!!

Note that the next claim of 1978 (with claim number 7800002) is congruent to 70,002 so is a synonym of 7100002 and also lead to a serial search which ends at location 130,002! Thus each successive DBPUT results in a search of 60,000 records 59,999 of which it had inspected during the preceding DBPUT!!

PAPA BEAR had claimed another victim!! The designer of this system had unknowingly laid a trap which would snap at a mathematically predictable time, in this case 1978. After struggling with this problem for months, the user ultimately escaped from PAPA BEAR by converting to "hashed keys" (in both the database and the application modules); a very expensive conversion!

Note that the problem was not a synonym problem in the sense that synonym chains were long nor was it a "fullness" problem since the master dataset was less than 57% full when PAPA BEAR struck.

The problem was due to the fact that the records were maximally clustered whereas DBPUT's space searching technique for masters is optimum only under (nearly) uniform distribution assumptions.

Note that the performance of DBFIND and DBGET was excellent since the maximum synonym chain length was 2.

Another much shallower pitfall would have been designed if, in the above example, the claim numbers had been of the form NNNNNYY with the same capacity of 370000. In this case, the performance of DBPUTs, DBFINDs and keyed DBGETs would all degrade over time but would never reach the disastrous level of the DBPUTs of the example. In this case, the degradation would arise due to the length of synonym chains and due to local clustering.

Note that this modest pitfall could be eliminated by changing the capacity, for example, to 370010.

Note however that this problem would still arise if the capacity were merely changed, for example, to 370001.

It should be apparant by now that designers may avoid the clutches of PAPA BEAR by carefully (mathematically) inspecting the consequence of the values of their choice of integer keys in relationship to their choice of master dataset capacity.

MAMA BEAR ... the SORTED PATH pitfall

My first live encounter with a misuse of sorted paths arose in 1975.

The facts surrounding this incident were told to me by Jonathan Bale who was still on the IMAGE project. Neither one of us remembers the exact numeric details so I have used poetic license by making up numbers which seem to be reasonably close to the actual ones involved in the incident.

The user had created a database containing one automatic master data- set and one detail dataset related by a 2-character key and where the resulting path was sorted by some long-forgotten field(s).

The user had written a program which read a record from an input file, added two blank characters to serve as the search field and then performed a DBPUT to the detail dataset. This was repeated for all records of the input file.

At the time that Jon received a phone call, the tape had not moved for around 10 hours and the program had already been running(?) for at least 30 hours.

On inquiry, Jon learned that the input file contained over 40,000 80-character records and that the user was using IMAGE to sort these records!

This is an extreme example of a sub-optimal use of sorted paths. To see this, it is important to know that when adding a new record to a sorted path, DBPUT starts its search for the appropriate point of insertion at the end of the chain and then searches the chain backward until it encounters a record whose sort field(s) value is not greater than that of the record being added.

For input records whose sort field values are randomly ordered, the expected number of records to be searched is one-half of the length of the chain. When the chain is 20 records long the search will cover 10 records on the average. When it becomes 30,000 long, the search will cover 15,000 records on average!

For a file with 40,000 records to be sorted into one chain the expected number of reads to cover all searches is approximately 400 million with the last record alone expected to take 20,000!

The blocking factor of the input tape was 200. No wonder the tape hadn't moved for 10 hours!!

To avoid the clutches of MAMA BEAR, avoid using sorted paths if the chains are very dynamic or very long. The more dynamic they are the shorter they should be and the longer they are the less dynamic they should be. The term dynamic is used here to refer to the relative frequency with which entries are added and deleted.

Contrary to the many warnings you may read against using sorted paths, there are occasions when their use is infinitely better than any other option.

HP's Corporate Parts Center in Mountain View used a sorted path in its back-order dataset. The search field was the part-number and the sort-field was a priority assigned by order-entry personnel in such a manner that the highest priority back-orders were at the front of the chain.

When new parts were received, a clerk at the receiving dock would enter the part-number and quantity at a terminal. The program would then perform a DBFIND with that part-number on the back-order data- set followed by a sequence of chained reads. For each record in the chain, a packing slip would be printed showing the quantity and destination and the record was then deleted. This process was repeated until the chain was empty or all received parts were accounted for. In the former case, an additional shipping slip was printed so that the remaining parts would be delivered to inventory.

This 'on-line' technique eliminated unnecessary shipment of parts to inventory, minimized parts handling, facilitated shipments and minimized errors.

Even though the chains were sorted, most back-order chains were either empty or had only a few entries so that adding new entries was never really slow.

Another, even more outstanding, use is available to order-processing systems where each sub-system (or part) in a master dataset is related to its components in a detail dataset by the part-number of the subsystem (or part). The component-numbers in each detail record are also present as part-numbers in the master dataset and each of these in turn may be related to other components in the detail dataset. In other words, the "parent-child" relation implicit in the concept of "component" is recursive.

The detail dataset here is related to the master via a parent-number field and is sorted by component-number. The fields of the record are ordered to take advantage of IMAGE's extended sort to include component option and quantity.

This "clever" design together with a recursive procedure enables the application to provide on-line single- or multi-level, fully indented, bill-of-material explosions with the components at each level in component-number and component-option order. No sorting is required and the performance of the explosion is limited by terminal speed.

Although many people may recommend that you avoid sorted paths, try implementing either of these applications without them. Response time would be somewhere between bad and disastrous!

There really is a place for network databases and sorted paths.

BABY BEAR . . . a discussion of PATHS

One of the reasons for defining a path is to provide rapid access to all of the records in a detail dataset having a common search field value.

In general, a path should be defined only if (a) it is necessary for the application or (b) its speed of access is better than a serial search and it is frequently used or (c) its speed of access is so much better than a serial search that it is cost effective even if it is seldom used.

Remember that each path you define causes additional overhead for DBPUT and DBDELETE and requires more disc space.

In considering frequency of use, remember that if you have 16 paths they cannot all be being used more than 6.25% of the time so that any arguments offered by the proponents of a path or paths should be evaluated in light of the fact that the sum frequency of use cannot exceed 100%.

Good uses of integer keys require the designer's awareness of the effect of the key values and the capacity on the address assignments made by DBPUT over the life of the application.

For certain applications, the use of sorted paths is not only highly recommended but may,in fact, be critical to success. The back-order application described earlier was implemented by Jonathan Bale in 1974 and the bill-of-material application was implemented by myself also in 1974. In both cases, sorted paths were a must.

As illustrated in the examples, sorted paths can provide benefits critical to some applications.

For instance, the application may not have to search the entire chain or it may simply be easier to program and/or marvelously faster as with the bill-of-material example mentioned above.

The overhead for paths mentioned in reference to DBPUTs and DBDELETEs is also proportional to their frequency of use. In other words, this overhead is less of a consideration for relatively static datasets than for relatively dynamic datasets. So additional paths for static datasets have less DBPUT and DBDELETE performance costs than on dynamic datasets.

Summary

In general, the rule for a path is: "When in doubt, leave it out.". If leaving it out proves to be a mistake, you can be sure that someone will call it to your attention and then (with the help of ADAGER) you may add it without impact on any application module. On the other hand, if providing it proves to be of little benefit, no one will tell you and removing it will undoubtedly have dire consequences on some application module(s).

Biographical Sketch

Fred White began his Programming career in 1957 as a scientific programmer. He programmed for 8 years in various user environ- ments expanding into commercial, management system and system programming areas primarily on IBM and Burroughs computers.

On August 1, 1965 Fred took a programming position with IBM where he worked on a multiprocessing text processing system which ran on modified 1440s or 1460s. During his 4 years with IBM he furthered his knowledge of file sharing and concurrency control.

On August 1, 1969, Fred joined Hewlett-Packard as Project Man- ager of what later became the MPE File System. He designed the account/group/user structure, capability classes, file codes and other features of that system.

Fred is best known in the user community for his involvement with IMAGE/3000 where he served as Designer, Programmer and Project Manager.

Fred left HP and has been working with Adager since November 1, 1981.

Note: Sorry about being so wordy. 26 years are hard to en- capsule (at least for me).

HP1000 Topic, To Be Announced

It was the policy of the Program Committee to allow speakers on the conference Program if and only if their papers were submitted in time to appear in these proceedings. For the most part we adhered to that policy. However, our first concern was the quality of the Program and the timeliness of its topics. Those concerns took precedence over all others. At the time the Conference Proceedings were being compiled steps were being taken to combine the HP3000 and HP1000 user groups. Members of both groups approached the Program Committee and suggested that in light of this consolidation movement, it might be appropriate to include at least one HP1000 oriented topic, the Committee agreed. The Committee also agreed to waive the requirement for a paper for inclusion in the Proceedings.

At the time of this writing no specific topic had been identified for the HP1000 talk. The Committee agreed to rely on the judgement of Phil Hardin, Chairman of the HP1000 User Group, in selecting an appropriate topic.

Signed:

IUG ANAHEIM
PROGRAM COMMITTEE

A DISTRIBUTED NETWORK FOR DEVICE-INDEPENDENT COMPUTER GRAPHICS

by Gary L. Koenig and Clifton Harald
NOI Systems

INTRODUCTION

The purpose of this paper is to describe an approach to implementing a distributed network for device-independent computer graphics software on the HP 3000. A distributed network is a system in which the various elements function as independent processes (sometimes on different computers), cooperating to solve a single problem. Device independence is a feature of a graphics software package that allows a single application program to draw to many different display devices. The discussion will emphasize a single graphics package, the DI-3000 software available commercially from Precision Visuals, Inc., but has broad application to other device-independent graphics application software.

The first section of the paper will provide an introduction to the principles of device independence, and will include descriptions of the primary functional components of device-independent graphics systems. A comparison will then be made between distributed and modular software networks, drawing on actual experience with the DI-3000 package. Originally designed for distributed network implementation, DI-3000 has most often been used in modular software networks in which all modules are simply program subroutines. Within this section, the disadvantages of using modular software networks on limited address machines like the HP 3000 will be addressed in detail.

An approach to implementing a true distributed network will then be presented, again using the DI-3000 software as a specific case study. This part of the paper includes publication of the results of testing recently conducted to determine optimal methods of interprocess communication. A description of the benefits of distributed networks will be presented, substantiated by the results of performance testing. The paper concludes with a discussion of the potential for expanding distributed networking to support multiple CPUs and highly intelligent devices.

DEVICE INDEPENDENCE

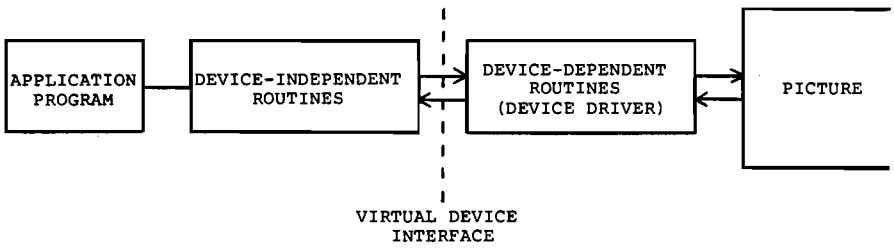
The objective of device independence is to enable a person to display the same or similar graphic images on many different graphics display devices. This capability is desirable for many reasons. Perhaps most importantly, device independence can minimize the effects of hardware obsolescence, since new display devices can replace outdated ones without changing graphics application programs. Secondly, graphics applications can be developed initially on inexpensive equipment and then, through device independence, transferred to more sophisticated devices for detailed modification. Additionally, device-independent software is not limited to a specific vendor's hardware inventory.

In a truly device-independent graphics system, general graphic requirements of a single application program are translated by software modules (called device drivers) into specific graphic commands for a target display device. Commands may be passed to the device as a string of ASCII characters, as binary information or in whatever form is required. Figure I depicts the primary functional components of a device-independent graphics system. An application program calls up device- independent routines which perform basic calculations and format an image in nondevice-specific terms. When the image has been properly formed, it is passed to a device driver through a virtual device interface. The virtual device interface functions as a communication link between the device-independent routines and one or more device drivers. Since each graphics display device has different capabilities and a different command language, separate device drivers are required for each device supported by the

graphics package. The device driver translates the image into device-specific commands which are then sent to the display device, resulting in representation of the image on the device.

FIGURE I

Device-Independent Graphics System



The key to device-independent graphics is the virtual display device, which is a hypothetical device that represents the combined graphics capabilities of all the devices actually supported by a graphics package. All graphics computations are oriented toward the actual device requirements represented by the virtual display device.

Since a virtual display device is an idealized representation of all graphics device capabilities, very few devices fully support the virtual device. Those functions that cannot be performed directly by the device must be either simulated or overlooked by the device driver. Capabilities such as line style, line width, polygon fill, polygon interior patterns, text attributes, and image transformations generally can be simulated. Color, intensity, real-time motion, and input capabilties generally must be overlooked if not specifically supported by the device hardware.

Device independence has been implemented in a number of different ways over the past decade. Current standardization efforts are attempting to provide a common basis for designing device-independent packages. One of the earliest set of standards was the Core package forwarded by the Graphics Standards Planning Committee of ACM/SIGGRAPH. The Core standard has recently given way to the international Graphical Kernal System (GKS) standard, which has been adopted by the ANSI X3H3 committee.

THE CORE STANDARD

Although the Core standard has been superseded by the GKS standard, it is still noteworthy, because a number of highly successful graphics packages are based on it. The Core system defines standards for the following graphics package features:

- o System and virtual device control;
- o Positioning and nontext primitives such as moves, lines, polylines, polygons, and markers in either two- or three-dimensional coordinate systems;
- o Attributes for positioning and nontext primitives. These attributes include color, intensity, linestyle, line width, polygon

edge style, polygon interior style and marker symbol;

- Text, text primitives and text attributes such as path, font, justification, size, gap, and base;
- Segments and segment attributes. A segment is a collection of output primitives making up a part or all of a graphic image. Segment attributes include visibility, highlighting and pickability, i.e., whether the segment can be uniquely selected from an interactive graphics device. Segments can be defined temporarily, or retained permanently;
- Transformations. Modeling transformations allow an object to be translated, scaled or rotated within its own coordinate system; Viewing transformations define the position orientation, line of sight and lens configuration that describe how the image will look when displayed on the graphics display device. Image transformations allow the displayed images to be translated, rotated or scaled on the display device surface;
- Virtual graphics input;
- Inquiry capabilities allowing the graphics system to determine the current state of all attributes and primitives.

One of the most intriguing challenges of implementing a Core-based system is to determine where to place the virtual device interface. Placement is typically defined in terms of its proximity to the application program. Figure II depicts a simple virtual device interface that is far from the application program. Figure III shows the interface closer to the program. As these illustrations suggest, the major complexity of a graphics software package can be found in either the device independent segment or the device driver segment of the system, depending on the placement of the virtual device interface.

FIGURE II

Simple Virtual Device Interface

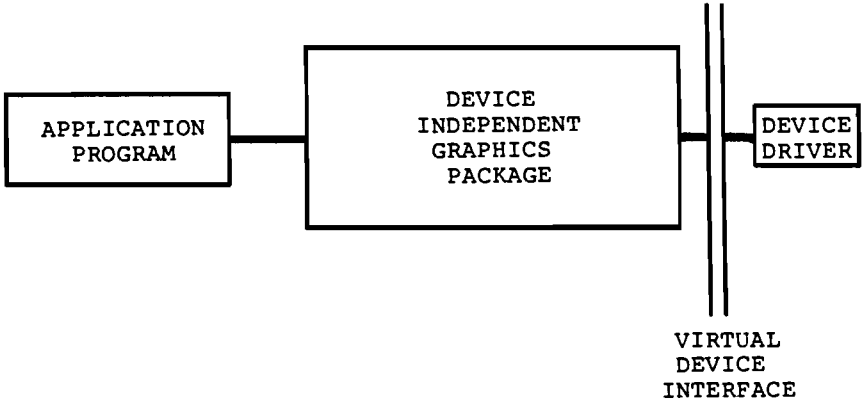
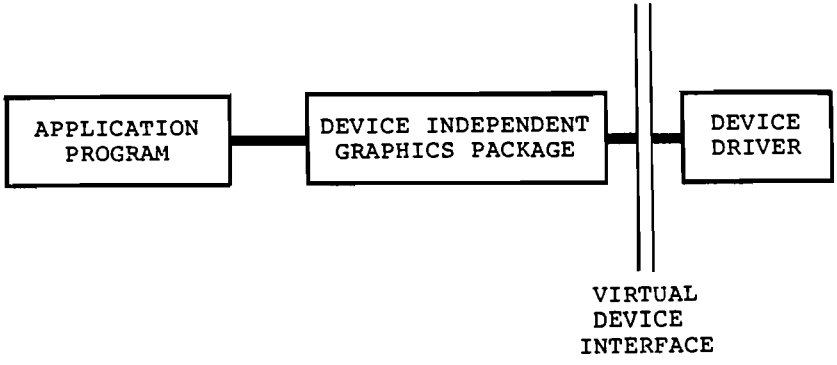


FIGURE III

Intelligent Virtual Device Interface



As the sophistication of graphics devices improves, it is generally felt that the virtual device interface should be moved closer to the application program, in order to take full advantage of the device features available. If the device has limited capabilities, the device driver will either simulate or override requests for functions not provided in the hardware.

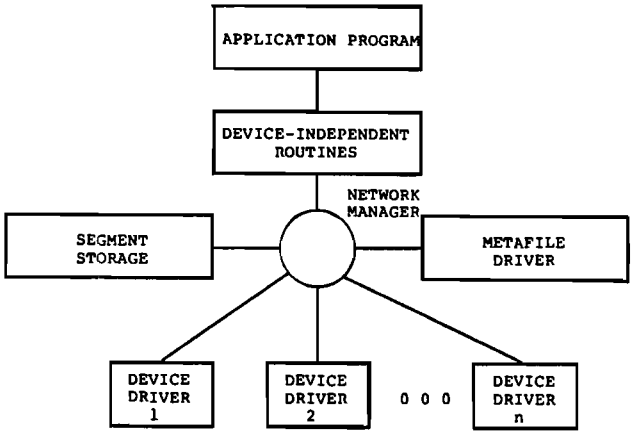
THE DI-3000 SOFTWARE NETWORK

DI-3000, developed by Precision Visuals, Inc. of Boulder, Colorado, is a commercially available system based on the 1979 Core standard. Designed as a complete three-dimensional coordinate system, treating two-dimensionality as a subset, DI-3000 is available in two configurations: Level A and Extended. Level A offers all the Core system features except for retained segment features. The Extended configuration adds retained segment capabilities to the Level A configuration.

DI-3000 was designed as a modular graphics software network. This approach lends itself to distributing tasks to intelligent graphics devices or among linked Central Processing Units. In this system the virtual device interface is located close to the application program, allowing DI-3000 to take full advantage of device capabilities. Figure IV illustrates the distributed network of DI-3000.

FIGURE IV

DI-3000 Distributed Network



The DI-3000 network is composed of four types of modules. In the first module, the application software is combined with the device independent routines. Device drivers reside in separate modules. Theoretically, multiple devices can be activated concurrently, and all devices can be selected at run time. The metafile driver module shown above is similar to a device driver, but "draws" to a disk file rather than to a display driver. These metafile images can be redrawn and/or modified at a later time by using a special metafile translator program.

All retained segments are stored in, and managed by, the segment storage module. Segment storage also provides support to the device drivers. Intelligent devices that maintain display lists (or segment information), do not need much support from the segment

storage mode. A command to make a segment invisible would be handled by such a device. A less intelligent device, however, would have to depend on the device driver and segment storage to erase, redraw or otherwise perform the software functions necessary to simulate segment invisibility.

As shown in Figure IV, the network manager is the hub of the DI-3000 network; all communication between nodes must go through the network manager. A sophisticated protocol has been developed to allow efficient message switching among the nodes. All messages sent through the network manager are made up of fifteen-bit, unsigned integers, or two eight-bit ASCII characters. In either case, the basic transmission unit is a sixteen-bit "word." The obvious advantage of this method is the ease of implementation on any computer that has at least a sixteen-bit word size. All of the modules in the network are tailored to the word size of the host CPU. The sixteen-bit restriction is in effect only for messages routed through the network manager. This approach enables a sixty-bit machine to host the application program and device independent routines while other nodes reside on a sixteen-bit CPU. The major disadvantage of the fifteen-bit resolution limitation is that it might be insufficient to drive ultra-high resolution devices.

COMMON IMPLEMENTATION OF DI-3000

Although designed as a distributed network, DI-3000 is most commonly installed as a subroutine library system, where all necessary routines (including the device driver), are bound together in a single program image. Although this approach is satisfactory for many applications, it does not take full advantage of the distributed features of the network. For example, device selection cannot be deferred until run time, because a device must be bound to the program at PREP time. Furthermore, since all routines are bound into a single program image, it is difficult to support multiple devices concurrently. Though typically used on the HP 3000, single bound program implementation presents difficulties due to HP architectural limitations.

The major concern in implementing DI-3000 on the HP 3000 arises from the product's memory requirements and the machine's memory management limitations. Figure V presents the code segment and data segment requirements for the DI-3000 routines. After these requirements are met, little data segment remains for the application program. As a result, the usefulness of a toolset product like the DI-3000 is restricted.

FIGURE V

DI-3000 Segment Requirements

	Approximate Data Segment Used	Approximate Code Segment Used
DI-3000 with HP7220 Device Driver	40K bytes	70K bytes
DI-3000 Extended with HP7220 Device Driver	47K bytes	102K bytes

The amount of code segment used raises special concerns. As noted above, DI-3000 is a library of callable subroutines, but the HP 3000 does not have well- developed library handling tools. Therefore, nearly all routines are bound to the application program, even if they are not required.

Neither RL or SL files provide an adequate solution to the code segment usage problem. Most DI-3000 routines rely on the use of COMMON (global) storage. As a result, they cannot be placed in an SL file. Although an RL file could be used for some routines, it would not serve them all, because routines loaded from an RL file are placed into the same code segment. As Figure V shows, the amount of code segment memory required would normally exceed the system limitation for a single code segment. If HP would allow multiple RL searches, however, the RL alternative would offer great promise.

Due to these limitations, most DI-3000 routines are placed into a single USL and the program is PREPed using the USL. PREP time is often longer than necessary, because many routines not required by the application program must be linked into the program image.

Since SL use is prohibited, very little code sharing is possible. Even though the code for the DI-3000 routines is identical for two different application programs, code sharing is impossible when the two programs are run simultaneously. Only when users are running the same program files can the advantages of code sharing be realized.

The Extended version of DI-3000 also presents limitations in usefulness. Since there is little data segment available for the application program, it is difficult to dedicate a significant amount of memory space to the segment storage module. An application program that has complex segment storage requirements is virtually impossible to create on the HP 3000 using bound program image implementation.

PROCESS HANDLING IMPLEMENTATION OF DI-3000

Since DI-3000 was designed for a distributed network environment, its implementation on the HP 3000 may be facilitated by using MPE's process handling capabilities. Implementation using process handling is straightforward due to the rigid separation between modules in the DI-3000 network. Since all communication between modules goes through the network manager, the device drivers, metafile driver and segment storage module may be regarded as "son" processes, invoked and controlled by the network manager. The network manager may then be bound together with the device independent routines and application program into a single program image.

Once the decision to utilize process handling is made, it is necessary to determine which interprocess communication alternative to use: message files, extra data segments or the RECEIVEMAIL and SENDMAIL intrinsics. The choice is not obvious, however, because the network is not being implemented for simultaneous processing purposes. Message files are the most reasonable alternative when the processes operate simultaneously, and must communicate with each other at irregular intervals. In DI-3000, however, the network manager passes a message to a device driver (or other "son" process), and then waits until a reply is sent back before attempting to perform any other function.

INTERPROCESS COMMUNICATION TESTS

To test the resources used by each of the communication methods described in the preceding section, several relatively simple FORTRAN test programs were designed. The "father" processes were coded as follows:

Message File	Extra Data Segment	MAIL
Find the time of day	Find the time of day	Find the time of day
Arbitrarily initialize a 256-word buffer	Arbitrarily initialize a 256-word buffer	Arbitrarily initialize a 256-word buffer
Build and FOPEN an input Message File	Get a 256-word Extra Data Segment	
Build and FOPEN an output Message File		
Create and activate the "Son" process	Create the "Son" process	Create the "Son" process

Message File	Extra Data Segment	MAIL
Send and receive 1000 256-word buffers using FWRITE and FREAD	Send and receive 1000 256-word buffers as follows: DMOVOUT the buffer; activate the "Son" process; suspend itself; when activated by the "Son,"	Send and receive 1000 256-word buffers as follows: SENDMAIL the buffers; activate the "Son"

	DMOVIN the buffer	process; suspend itself; when activitated by the "Son," RECEIVEMAIL the buffer
Find and display the "Father's" CPU time	Find and display the "Father's" CPU time	Find and display the "Father's" CPU time
Find and display the Elapsed time	Find and display the Elapsed time	Find and display the Elapsed time
Exit	Exit	Exit

In all cases, the "son" processes essentially performed mirror actions. The programs were run on a Series 40 with 1 megabyte of memory and two Winchester drives on a single GIC. One set of tests was run on a stand-alone machine, and another on a machine that had reached steady-state conditions under a heavy, reproducible load. The latter tests were designed to heavily stress the CPU, memory and disc channel.

Figure VI presents the results of running each test five times under no-load and heavy load conditions. The tests consistently demonstrated the advantage of using the extra data segment method. Although the coding of this method is slightly more difficult than the coding of the message file method, the inconvenience is compensated for by lower execution and elapsed times. As a result of these tests, DI-3000 was implemented as a process handled network, passing messages through a 256-word extra data segment.

FIGURE VI

Interprocess Communication Test Results
(All times in seconds)

NO LOAD

	Message File			Extra Data Segment			MAIL		
Test	"Son" CPU	"Father" CPU	Elapsed Time	"Son" CPU	"Father" CPU	Elapsed Time	"Son" CPU	"Father" CPU	Elapsed Time
1	1.50	1.76	8.19	0.71	0.84	5.07	1.40	1.50	16.38
2	1.47	1.85	8.26	0.71	0.84	4.88	1.43	1.54	16.44
3	1.46	1.85	8.27	0.72	0.84	4.84	1.44	1.54	16.39
4	1.47	1.85	8.21	0.72	0.84	4.81	1.44	1.53	16.39
5	1.45	1.86	8.18	0.71	0.84	4.82	1.42	1.54	16.41
Average	1.47	1.83	8.22	0.71	0.84	4.88	1.43	1.53	16.40

WITH LOAD

1	1.43	1.76	10.28	0.71	0.82	7.49	1.50	1.62	19.71
2	1.44	1.79	10.47	0.71	0.82	7.49	1.60	1.68	19.64
3	1.44	1.78	10.60	0.69	0.82	7.32	1.58	1.68	20.92
4	1.42	1.79	10.49	0.70	0.81	7.37	1.58	1.68	20.50
5	1.43	1.79	10.36	0.71	0.81	7.36	1.59	1.70	21.55
Average	1.43	1.78	10.44	0.70	0.82	7.41	1.57	1.67	20.46

BENEFITS OF THE PROCESS STRUCTURE

By using HP's process handling capabilities, the advantages envisioned in the design of the network have been realized. Furthermore, most of the difficulties arising from HP hardware and software limitations have been overcome. Through process handling, for example, graphic devices can be selected at run time. Device drivers are not bound to the program image, since they run as "son" processes. A special routine was added to DI-3000 to allow the programmer (and ultimately, the user) to decide which device driver to activate at any time during program execution.

Using process handling, drawing to more than one device at the same time presents no special problems. Although DI-3000 was designed to support multiple devices concurrently, single bound program image implementation practically precludes use of this feature. This is because many of the same routine and common block names are used in all device drivers. Since no device drivers are bound in the process handling implementation, users of the network are not limited to a single device driver.

It is important to remember that a "son" process must always respond to the network manager before further processing takes place in the device independent part of the network. This allows use of the same extra data segment to communicate with all "son" processes. Consider, for example, a program that draws a pie chart to an HP2623 terminal and an HP7221 plotter simultaneously. The device independent routines form a slice of the pie in a virtual device form. The network manager activates the HP2623 device driver and sends a message through the extra data segment. The HP2623 device driver translates the message, draws the slice on the screen, and then replies to the network manager. The network manager then activates the HP7221 driver and sends the same message through the same extra data segment. The HP7221 driver then translates the message, draws the slice and replies to the network manager. This same sequence is repeated until the entire pie chart has been drawn on both devices.

Another benefit of process handling is a large increase in the amount of data segment available to the application program. This is a result of all global and local storage in the device drivers, metafile driver and segment storage being moved out of the "father" data segment and into data segments for the "son" processes. DI-3000 now takes approximately 20,000 bytes of data stack for both Level A and Extended versions. This requirement is a substantial improvement over the 40-47,000 bytes required when everthing is bound into a single program image. As a result of increased data segment availability, the programmer not only has more data stack, but considerably more segment storage, as well. Segment storage is a separate process with its own data segment, which allows programmers to dedicate more than 25,000 words to segment storage.

Under process handling implementation, significant improvements in the efficiency of program creation are realized. The time required to PREP an application program is diminished, as is the complexity of the operation. USL management is more efficient, because the USL does not contain code from device drivers, segment storage and the metafile driver. Similarly, the Segmenter takes less time to PREPARE a program because it has fewer routines to link. The only detrimental aspect of process handling is an increase in drawing time. Although the increase has not been measured and will vary with the baud rate and display device, it is expected to be no more than ten percent in the worst case.

One of the most important benefits of process handling is that all "son" processes may be shared, since the program image of these processes does not change from application program to application program. Sharing of "son" processes thus relieves some of the strain that DI-3000 can place on memory.

A potential benefit that has not yet been fully implemented is the ability to call DI-3000 routines from COBOL, BASIC or PASCAL. Almost all FORTRAN input and output takes place in the device drivers and the metafile driver. The remaining FORTRAN input/output mainly addresses error and debug processing. Once the latter input/output is transferred to a "son" process, or converted to system intrinsics, DI-3000 should be available to host languages other than FORTRAN.

FUTURE POSSIBILITIES

Two extensions of process handling principles offer potential benefits in future applications: concurrent multiple processing, and distributed processing using more than one computer.

Concurrent multiple processing offers the most potential in an environment where many different DI-3000 application programs access a large number of devices. In such an environment, the optimal system configuration would most likely call for separation of the network manager from the device-independent module, placing it in a global process that communicates with several application programs, as well as all device drivers, segment storage and the metafile driver. The application programs would asynchronously send messages to the network manager to be forwarded to the appropriate driver or segment storage.

The network manager would thus direct the flow of signals between all programs and processes.

Distributed processing offers even more potential, given the increasing intelligence of graphics devices. A network in which device drivers actually execute functions within specific devices would lighten the load placed on the CPU and the main memory. It is significant that a device driver that runs and draws on the IBM PC is currently available. True distribution of processing will continue to prosper as more graphics devices gain a high level of intelligence.

SUMMARY

Device independence is a very desirable, perhaps indispensable, feature for any graphics package to possess. Standards have grown up around this feature and have, in turn, been implemented as commercial packages.

DI-3000, the package described in this paper, was designed as a software network. Performance problems arise when DI-3000 is simply installed as a library of subroutines. Taking advantage of the modular design by implementing DI-3000 as a process handled network releases its full power while minimizing its effect on system resources.

The network design can be extended to allow some processes to run on separate computers, especially intelligent graphics devices. This is a practical and efficient example of a true, distributed processing network.

BIBLIOGRAPHY

Olenchuk, Bruce. "Graphics Standards." Computer Graphics World; Volume 6, Number 8. August 1983.

Precision Visuals, Inc. DI-3000 User's Guide. Boulder, CO: Precision Visuals, Inc.

Warner, James R. "Principles of Device-Independent Computer Graphics Software." Los Alamitos, CA: The Institute of Electrical and Electronics Engineers, Inc., 1981.

BIOGRAPHICAL SKETCHES

Gary L. Koenig is a computer consultant for NOI Systems of Boulder, Colorado. He provides a wide range of services for HP 3000 sites, but specializes in computer graphics, accounting and productivity tools. In the past two years, Koenig has supervised development projects utilizing many of HP's development tools: FORTRAN, COBOLII, PASCAL, and VPLUS.

Koenig's past experience includes system programming, DP planning, Data Center management, customer support, product planning, system performance, computer networking, and application programming.

Koenig received a BS in Mechanical Engineering from Iowa State University in 1969. He worked briefly as an application engineer before joining the Data Processing profession.

Clifton Harald is a Research Associate with the National Institute for Socioeconomic Research (NISR) in Boulder, Colorado. He specializes in evaluating the public cost and revenue consequences of community development trends, and in

analyzing alternative local government fiscal management policies. He has managed several socioeconomic impact assessment projects for NISR, and directs marketing of NISR's Planning and Management Services among local governments in the western United States.

Harald also has experience in energy and environmental planning, having previously worked as an Energy Specialist with the cities of Seattle, Washington, and Boulder, Colorado. As an Environmental Planner with the regional transit agency in Seattle, he coordinated facility planning for multi-million dollar construction projects and prepared technical analyses of transit system energy use.

A graduate of the University of Washington, Harald holds a Masters degree in Urban Planning. He received his Bachelor of Arts degree in Literature, Sociology and Psychology from the University of Colorado.

MPE log files for performance, security and debugging.

by Denys Beauchemin

1) Introduction

This paper will cover the different logging done by MPE on your behalf. We will review these log records, what they mean, and how to use the log records to analyse performance, help with system security and also on some small aspects of debugging.

We will examine a few true-to-life cases and try to set ourselves up for more control on the HP3000.

The following is a layout of the paper:
i) Log records, different types of.
ii) Impact of logging.
iii) Performance analysis with MPE logging.
iv) Enhanced security with MPE logging.
v) Debugging with MPE logging.
vi) Conclusion.

A final note before we get into the subject matter, you will find numerous references to a program called LOGANAL. This program is on the swap tape and is the primary tool used in the analysis of log files. The program was written by the author of the paper and contributed to the IUG thru the swap tape mechanism, and as such no implied or explicit guarantees and/or responsability is assumed by either the IUG and/or the author and his employer.

I) LOG RECORDS, DIFFERENT TYPES OF.

As of the CIPER mit, the following events could be logged by MPE, provided that these events were 'turned on'. For those of you who may not know, you 'turn on' the events at COLDLOAD time either from a tape created by SYSDUMP or on prompts from INITIAL.

Event	Description	
0	LOG FAILURE	Log failure record
1	HEAD RECORD	First record when system comes up
2	JOB INITIATION	Logon record
3	JOB TERMINATION	Logoff record
4	PROCESS TERMINATION	Process termination record
5	FILE CLOSE	File closure record
6	SYSTEM SHUTDOWN	System shutdown
7	POWER FAIL	Power failure record
8	SPOOLING	Spoolfile creation, mod, destruction
9	LINE DISCONNECTION	DSN line disconnect, DS, RJE, MRJE etc
10	LINE CLOSE	DSN line closes, DS, RJE, MRJE etc
11	I/O ERROR	Any error on any devices
12	VOLUME MOUNT	Physical private volumes mount/dismount
13	VOLUME SET MOUNT	Logical private volumes mount/dismount
14	TAPE LABELS	Tapes labels recognized by AVR
15	CONSOLE	Console log event record
16	PROGRAM FILE EVENT	Underflow simulation record
17	CALL PROGRESS SGNLS	
18	DCE PROVIDED INFO	

0- LOG FAILURE.

This record is written as soon as MPE logging is reeabled. It contains the following information. No of log records, no of logons & no of logoffs missing. MPE logging can be stopped for various reasons, but when it is to be restarted, the following command should be issued: RESUMELOG. You should consult the console operator guide for the causes of log failures.

1- SYSTEM UP.

When the system is brought up, this record is written to the log file. It contains come coldload information such as:
Update level
Fix level
Core size, main memory
size expressed in Kwords.
CST size, number of entries in CST.
DST size, number of entries in DST.
PCB size, number of entries in PCB
IOQ size, number of entries in IOQ.
TRL size, number of entries in TRL.
IDC size, number of entries in ICS.
Maximum number of running jobs/sessions.

2- JOB INITIATION.

Every time a session or job is started, this record is created. It contains the following data.
Job type, Job number.
User name, account name,
job name and logon group name.
Input device number and
output device number.
Logon queue, B,C,D,E.
CPU time limit, for the job/session.
INPRI and OUTPRI.
LOGANAL makes use of this record, especially for security purposes. It will also be usefull to you for performance recording.

3- JOB TERMINATION.

When a session/job is terminated, this record is created. It contains the following info.
Job type, job number.
Maximum priority, ever attained
by any process, usually 0.
Number of creations, of processes.
Ie of many programs runned.
CPU time, in seconds.
CONNECT time, in minutes.
LOGANAL will report on this. It is very usefull when you are tracking a specific session/job.

4- PROCESS TERMINATION

This record is created for 3 events. First, when you terminate a program, second, if you have a program that has launched a process, and this process terminates. And third when you log off, don't forget that a session/job is a process. Here is the data contained in the record:
# of program file segments.
number of sl segments,
non-mpe this impacts your CST.
maximum stack size ever, expressed in words.
maximum data segment size ever,
expressed in words. This is for any
extra data segment.
cumulative total of virtual storage,
ammount of disc in sectors
requested for data both stack
and extra data segments.

5- FILE CLOSES

Every time you open a file, it will be closed either by yourself or by MPE. This record is created at file closure time. This record is usefull in security tracking and also is used for performance analysis. It is also used for debugging, but more on that later. The information is the record is as follows:
File name, fully qualified.
Disposition, of file at fclose, ie delete, save, etc.
Domain, of file at fopen,
ie new, temp, perm etc.
Number of sectors allocated,
sectors reserved on disc.
Device type, on which file resides.
Device number, on which file resides.
Records processed, since the fopen for that file.
Blocks processed, since the fopen for that file.

6- SHUTDOWN

When a =SHUTDOWN command is executed, this record will be written and contains the following info.:
number of jobs, on system at shutdown.
number of sessions, on system at shutdown.

7- POWER FAIL

This is self-explanatory, and the record contains a power fail recovery status flag (it should be set at 0, but I have seen it a %177777 (-1)) (documentation problem?).

8- SPOOLING

Every time a spoolfile is created, modified or deleted, this record is created. Since I have not used this feature, I will not discuss it any more than to say what information is recorded:

User name, account name, job name, file name.
Job type, comes from either a job or a session on this, or another system.
Job number, of original creator.

I/O, 0=input and 1=output.
Devicefileid.
Device type, 0 for input and 32 for output.
Spoolee number, input=disc number on which file resided,
output=device number printer upon.
Numcopies, number of copies yet to be printed.
Number of records processed, input-number of lines input
output-number of lines printed.
Number of sectors used, or occupied on disc by the spoolfile.
Subtype, input-0, output=subtype of printer used.
Func, is the last operation of the spooler
normal completion, deletespoolfile, defer spoolfile, relink.
Number of physical pages, only for 2680a laser.
Number of Logical pages per physical pages, only for 2680a laser.

9- LINE DISCONNNECT

When you have a DSN link that disconnects, this record is created. Again, I have not used this event so here is just the information.
Logical device number.
Time of connection.
Number of output data transfers.
Number of input data transfers.
Number of recoverable line errors.
Number of irrecoverable line errors.
Local ID sequence.
Remote ID sequence.
Phone number of remote, if local system dialed.

10- LINE CLOSES

This record is created when the line is closed. I have the same things to say about it as in number 9. Here is the record format:
Logical device number.
Time stamp of open.
Driver name, max 8 ascii digits.

11- I/O ERRORS

This record is very important when time comes for a PM. It is created for every occurance of an I/O error on any device on the system. I will not cover the data at all.

12- VOLUME MOUNT

This record is created every time you physically mount a pack on a private volume drive. Since I have never used private volumes, I will not cover the data at all.

13- VOLUME SET MOUNT

This record is created when a user requests the use of a pack. See 12 above.

14- TAPE LABEL

This record is created every time a tape is mounted on a drive and AVR (automatic volume recognition) identifies it as a label tape. I will not cover the data.

15- CONSOLE LOG

This record is created by many different events occuring at the console. It may record action from or action directed to, the console. I will try to present a list of the events recorded. But first, here is a layout of the information contained. Byte length, of console line, length is negative for input, positive for output. Console line.

16- PROGRAM FILE EVENTS

This record came about with the advent of MPE IV. It records the underflow situation simulated by MPE. The most common example is for BASIC programs compiled under pre-MPEIV MITs that contain the CHAIN or INVOKE command.

17- CALL PROGRESS SGNLS

I have no information on this event.

18- DCE PROVIDED INFO

I have no information on this event.

There have also been at one time, and may still be 2 other events that could be logged: 46-MPE maintenance request log and 47-DCU (diagnostic control unit) log. I do not know exactly what these are for.

II) IMPACT OF MPE LOGGING

In order to analyse log files to get information on performance and security, you must enable the following logging:

Type	Reason
2	Job/Session logon records
3	Job/Session termination records
4	Process termination records.
5	File close records

Warning: When type-5 (file close) is enabled, your log files will jump in size very rapidly.

You should dump these log files on a very regular basis and then put them on tape for later analysis, then purge them.

III) PERFORMANCE ANALYSIS

If you were to look at the logfiles with LISTLOG2, you would rapidly see that you are unable to get a clear picture of the events easily. This is why I created the program LOGANAL which you will find on the swap tape.

From now on, we will only look at logfiles as seen thru LOGANAL. We will not bother with LISTLOG2 except to mention that a number of events are disregarded by LOGANAL (ie, power fails, I/O errors etc.), I feel that LISTLOG2 does a good job of reporting those.

Now, here is an example of a typical short session:

INSERT EXAMP1 HERE (*)

Here is also another small view of a file used during that session:

INSERT EXAMP2 HERE (*)

As you can see, this is a blow-by-blow account of the session from beginning to end. Let's look closer:

We have the console logon message, followed by the type-2 job init. record. Next we have a type-5, file close on COMMAND.PUB.SYS, this file is used to find out which UDC's are to be set for the session.

Then I went in to TDP, and you see that the program file TDP.PUB.SYS has two fcloses, a system fclose and a session fclose. I suspect that the loader does strange things to a program file when you run a program. Then the file TDPPARMS.TDPDATA.HPOFFICE is closed, this file is accessed by TDP when you first run the program, it sets up all sorts of parameters. It then closes PARMSET in my account because I have my own parameters.

I then texted in a file called PAPERC and you can see the fclose for that file after TDP opened for some checking thru FGETINFO I suppose. TDP also close the keep file K3480924 as new permanent file. And then it closes PAPERC as and old permanent file after copying the contents over to the keep file.

Then I keep the file back as PAPERC, and you can see the file PAPERC closed with delete option. Then PAPERC is again closed as a new permanent file after TDP has copied the data from the keep file. And finally K3480924 is also closed with delete option.

I then exit TDP and you see the type-4, process termination record, and a series of fcloses for $STDIN, STDINX and 2 $STDLIST.

The next program that I ran was a BASIC compiled program XDPROG. At the end of the program, you see the type-4 record, followed by the standard $STDIN, STDLIST and 2 new ones BASLIST and BASIN. These last files are used only with BASIC compiled programs.

And then I terminate my session, so you see a type-4 record for the session. Don't forget that a session is an MPE process! If you don't believe that, just do a SHOWQ and look at all the Mxxx PINs. You then see the fcloses for my UDC files, and finally the console logoff message and the type-3, job termination record.

I have added an extra line of report in LOGANAL, this tells you how many files the session used and the maximum and average values for the following: Stack, Data segment and Virtual storage.

Another feature of the program is that you can specify a certain file, and it will look for each and every occurance of a file close for that file. This is a good tool for security purposes, since there is no way that a standard user can access a file without a log record being produced. Note that if one were to use DISKED2.PUB.SYS, the logging can be circumvented.

System load is defined as who is doing what to which file and at what time. This means that we want to know if there are certain times in the day when the system is more active or less active than at other times.

As an example, we could see that the response time during lunch is much better than say 10:00am. These you can determine by yourself, or you can get these thing out of the log files.

One method is as follows:

You pick a certain day and you take the corresponding log file.

You then do a listing of the logons and logoffs for that period.

You then produce a listing of the activities of each session.

You can then plot on paper which programs & which files were accessed and by whom during that period.

Now you, being fairly knowledgable about your system, can then be in a position to determine which processes were using the most resources, which processes were superflous, which ones could have been rescheduled and which ones could have be deleted outright.

Of course, if you were to do this for any extended period, you would start being flooded by information, but one thing you may do is to modify the program to do the above for you.

We will look at a specific example at the conference.

We can easily see the periods of intense activity and the slow or slack periods.

We can also see that if we have UDC's, that just by logging on and off, we generate a series of fopens and closes.

We can also see some limitations with this method, First, it does not give us a perfectly exact picture of the system activity for a very specific period of time, but if does give us an overall look into the workings of the system. It provides wonderfull tracking. Second, all this data can only be analysed after the fact, but again, if we do this exercise for an extended period on a typical environment (mix), one will be able to ascertain the peak and slack periods very easily.

IV) ENHANCED SECURITY WITH LOGGING

I will relate a true-to-life incident that occurred some months back, and at another company.

We had on a system a 'GOD'-like or SM program, a utility that, temporarly, gives the user all capabilities. Now, to be useful, this program had been renamed, released and lockworded.

Only a select, ie very small group, knew that it existed and were supposed to access it. At one point, I heard that some other users knew about and had been using this program.

So, I decided to find out for sure. I ran LOGANAL and got a list of all fcloses on that program for a number of weeks in the past. There were something like 12 fcloses. Then I ran a list of logons & logoffs for the same period. I then compared the session numbers of the fcloses to the logons. Rapidly I ascertained that 11 or the 12 fcloses were from legitimate users, (namely myself). But the last one, I could not remember executing. So I took a listing of the session in question, and came up with a very interesting listing. The session initiated after hours, and on a secluded terminal. I knew who the user was. The user ran TDP, did a few things to his/her files, then terminated TDP. Then the user ran the 'GOD' or SM program 15 minutes into the session. The user proceeded to look at a few innocent files, then went into FCOPY. At which point he/she proceeded to look at some very confidential files.

You can, with this method, follow any job or sessions, you can also find out all the users who accessed a specific file or file range. This can be a great help in documentation.

V) DEBUGGING WITH MPE LOGGING

Again, the best way to explain this feature is to relate a true experience. One day, while near the console, I saw the following message: LOGxxxx 1/2 FULL. That's quite common, but what followed is not, because 30 seconds later I saw: LOGxxxx 3/4 FULL, and soon after: LOGxxxx FULL, LOGxxxx+1 IS ON. Something was wrong, some event was being logged at an incredible rate. I started LISTLOG2.PUB.SYS, thinking that it might be an I/O problem of some kind, and LOGANAL does not report on those. On the laser printer I see a long series of file closes coming from a job, and all on the same file. 'Abort job so and so' I shout to the operator. This done, I recognized the file name, it was used only in a small SPL routine that places a program in a wait state for data coming back thru MRJE. I had tested that routine time and time again, and it always worked perfectly. But the data coming back thru MRJE was always relatively short, except on that day. There was a small bug in the routine; If the file checked was busy, the procedure would start over again without going thru the waiting stage. This never impacted on a short transmission, but wreaked havoc on long transmissions. This is the kind of bug that will usually leave no trace. The fix took all of 4 seconds wall time. Without proper MPE logging, we might never have spotted this bug, and we might have looked elsewhere for the reason of very increased activity while waiting for MRJE data.

With MPE logging and LOGANAL, you can track a complex program of going thru many files, and thus assure yourself of proper functionnality.

VI) CONCLUSION

In this paper, we have looked at three different uses of MPE logging, Performance analysis, security and debugging. MPE logging also gives the C.E.s an idea about which devices may cause trouble. I have also seen MPE logging used for a job accounting environment. There are many uses one can make of the data contained in the log files. After all the data is there and the mechanism to capture it is part of MPE so let's use it!

The material on logfile record layouts was taken from the system manager/system supervisor manual, from Hewlett-Packard Ltd.

(*) There were no insert examples supplied.

A MANAGEMENT OVERVIEW OF COMPUTER POWER PROBLEMS

by ED MUXO - PRESIDENT
COMPUTER POWER SOLUTIONS, INC.

"Power Problems" are probably the most mysterious and misunderstood problems which can affect the Data Processing user. This guide is presented to assist the DP manager in preventing power problems from interrupting operations.

The first portion of this article defines, "What is a power problem and how do I know if I have one?" The second portion reviews Hewlett-Packard power specifications for the 3000 family. The final portion addresses the resolution of those problems, along with an analysis of the different types of power conditioning equipment.

How can you tell if you have a power problem? There is no way to "see" a transient, a sag, a surge, or a frequency deviation. The only kind of power problem you can see is a power interruption (blackout). Unfortunately, blackouts do not cause the majority of "power problems".

The following questions reflect the classic effects of power and/or grounding/ wiring problems.

1. Do you have one device that always seems to be down, and does not seem as if it can be fixed?
2. Is system reliability significantly lower than expected?
3. Do you have "intermittent software problems"?

There are four generic categories of power problems. These problems are (1) Transients, (2) Sags or Surges, (3) Blackouts, and (4) Frequency Deviations.

All power systems have transients. A transient/impulse/spike is a short duration, high amplitude pulse, superimposed on the power sine wave. Transients are the most common type of power problem. Transients may be generated by improper grounding. A solid, computer-grade, single-point ground is a MUST for reliable system operation.

Transient/noise problems can be solved by putting the computers on individual circuits which have no noise-producing loads (dedicated circuits), by decreasing system sensitivity to power problems (with the vendor including additional power conditioning devices within the computer system), or providing some type of isolation from impulses on the power lines (power protection equipment).

Sags and surges are changes in the am of the AC sine wave. They may la several cycles to several hundred Voltage sags can come from external

(the power utility "browning out" its users to help during power shortage periods), or internal sources (most commonly, an overloaded power panel).

A brownout is a scheduled long-term voltage sag. Brownouts force power supplies to draw more current. This can cause overheating and component stress. Brownout conditions may also cause certain devices to appear more sensitive to transient impulses.

Regardless of the origin of a voltage drop, short-term voltage fluctuation problems may be difficult to identify. Voltage dips may cause portions of the system to execute a pseudo-power fail routine that the operating system software cannot handle, with unpredictable results. High voltage surges can cause equipment damage, in addition to "baking in" long-term problems.

Frequency deviations are a shifting of the basic powerline frequency of 60 cycles per second. Frequency deviations are usually not a problem in the United States when using utility power.

Blackouts are complete voltage interruptions. These power discontinuities may last from sub-cycle events to seconds/minutes/hours/days. Blackouts can cause significant hardware damage because of the very large transients and voltage variations that can occur when the power grids are broken and re-established. Blackouts are usually never "clean".

Hewlett-Packard has incorporated circuitry within the CPU to detect an impending power failure and enter a power fail routine. When power is restored, the system will re-boot and processing will resume. During a "dirty" or miltiple power fail, the CPU may enter and exit the power fail routine several times. The software routines may not run to completion, and the system recovery attempt will fail.

Extremely large transients may be generated when power grids come on/off line. There may be significant voltage oscillations until the power grid is positively re-established. Power protection in areas subject to frequent blackouts or voltage breaks is a must. Both voltage and transient protection need to be provided. Larger systems users may front-end their systems with a motor-generator set, uninterruptable power system, or backup power generator to prevent hardware problems carried with blackouts.

*****

System grounding is very important to ensure a properly functioning system. The ground wire serves two purposes--as a safety path to ground and as a zero signal reference source for all the digital logic. No computer system can operate reliably without a solid, low impedance (AC + DC resistance) ground.

If the only concern were safety, any metallic path leading to ground would be satisfactory. Low-level, high-speed computer logic circuits require a solid electrical ground. There should be no short-cuts or com- promises when implementing computer system grounds. Building conduit, although it meets the requirements of the National Electric Code, is not reliable for use as a computer system reference ground.

If building conduit is the only ground provided for the system, the qual- ity of that reference will depend upon how tightly the conduits are mechanically connected. Any change in the integrity of those connections because of age, building movements, heat or cold, can directly impact the reference (and indirectly the safety) ground. These events may impact reliable computer operation.

There are three reasons for grounding electrical systems:

1. To maintain any non-current carrying exposed metal surfaces (conduits, equipment enclosures, etc.), at 0 volts, or ground potential with respect to earth - (PERSONNEL SAFETY/PROTECTION).

2. To provide an intentional path with ample fault current capacity to cause fuse or circuit breaker operation during a short-circuit condition - (EQUIPMENT SAFETY/PROTECTION).

3. To contribute to proper performance of the electrical system and equipment - (EQUIPMENT PERFORMANCE).

The five components of a grounding system are:

1. GROUNDED CONDUCTOR. This conductor is intentionally grounded. This is the neutral conductor (white wire). It carries return current back to the source.

   The neutral has two purposes: to permit utilization of power at the line-to-neutral voltage, and to provide a low impedance path for fault currents to return to the transformer neutral, thereby quickly opening a circuit breaker during a fault condition.

2. EQUIPMENT GROUNDING CONDUCTOR. This conductor is used to connect the non-current carrying metal parts of equipment, conduits, and other enclosures to the system grounded conductor and/or the ground- ing electrode conductor at the service entrance or at the source of a separately derived system. This is the "ground" (green wire). It will normally never carry current, except during a system fault con- dition.

   The ground has two purposes: to maintain 0 volts with respect to ground on the equipment enclosures during normal operation, and to provide a low impedance path for fault currents when a phase to ground fault occurs. The National Electric Code addresses itself primarily to safety, and does not specifically address situaions to reduce noise which could impact computer operations. Recent changes to NEC Article 250-74, Exception #4, have attempted to integrate safety and equipment performance.

3. GROUNDING ELECTRODE CONDUCTOR. This conductor is used to connect the grounding electrode to the equipment grounding conductor and/or the grounded conductor at the service entry or at the source of a separately derived system. This is the conductor which connects the ground/neutral bond to the building ground or the ground stake. It will normally never carry current, except during a system fault condition.

   The purpose of the grounding electrode conductor is to provide a low impedance path to connect the elecrical system to earth through the grounding electrode and also reference the neutral and ground wire to the same potential.

4. GROUNDING ELECTRODE. This item carries practically no fault current. During a fault condition, current flows back to the service transformer via the service neutral to complete the circuit. This is the building service entry ground, or a ground stake.

5. MAIN BONDING JUMPER. This is one of the most important connections in the entire power system. This connection is made between the grounded circuit conductor (the neutral) and the equipment grounding conductor. This jumper MUST be present at the main service breaker and the secondary of all transformers, especially isolation trans- formers.

*****

HOW CAN YOU DETERMINE IF YOUR SYSTEM HAS A POWER QUALITY PROBLEM?

Special devices are available from either Hewlett-Packard, consultants or power conditioning equipment vendors to monitor power quality. The generic term for these devices is "line disturbance analyzer".

High-speed line disturbance analyzers are the proper devices to use to qualify sag/surge/transient problems. These devices can quickly respond to short-term sags and surges and have the capability to "capture" transient impulses. These analyzers can also provide printouts of voltage abnormal- ities and summaries.

Because of their relatively slow response, the strip-chart recorders used by the power company are usually of limited help to identify short-term variations. They may be helpful in identifying long-term brownout conditions.

The first step in determining if you have a power problem is an in-depth review of the system wiring and grounding. If there are wiring or ground- ing problems, analyzers may falsely report conditions which are not present.

After the wiring and grounding have been verified, the next step is to con- nect a power-line monitor to your system power system to determine if your power quality is within the Hewlett-Packard specifications.

*****

WHAT ARE THE HEWLETT-PACKARD "POWER" SPECIFICATIONS, AND HOW DO THEY RELATE TO THE POWER QUALITY AT YOUR INSTALLATION?

HP has published power specifications for the 3000 family in the site prep manual. The system specifications are as follows:

1. VOLTAGE SPECIFICATIONS -

   a. SYSTEM III, 30, 33, 40 and 44: Between 108 (-10% low) and 124 (+4% high) as referenced to a nominal input voltage of 120 volts.

b. 3000/64: Between 187 (-10% low) and 220 (+6% high) as referenced to a nominal input voltage of 208 volts.

If your input voltage remains within these two limits, you are operating within the HP voltage requirement specifications.

2. TRANSIENT IMPULSES - LESS THAN 100 VOLTS

If the transient impulses (spikes on the powerline) are less than 100 volts, you are within specifications.

One important exception - the HP64B contains an integrated conditioner. Because of the integrated conditioner in the CPU, you can obtain significant installation cost savings by not having to purchase three-phase power conditioning equipment to protect the CPU. Only the peripherals require protection.

"-db measurements" are a method of specifying transformer high frequency noise rejection characteristics. The higher the "-db", the greater the noise attenuatin capability of the transformer. A -60db transformer seeing a 20KHz, 1 volt spike in, will output a .001 volt spike. A 1000 volt transient will result (ideally) with a 1 volt output at the secondary. Note that these are only mathematical calculations, and usually have NO actual relationship to what the system may actually see under a live load. Proper wiring and instal- lation are a must to ensure correct power conditioning equipment performance.

Understanding the specifications and how they specifically relate to you and your system can prevent many problems. Providing proper wiring, grounding, and in-spec power is your responsibility. Provid- ing the information you need to ensure that the specifications are met is HP's reponsibility.

Service engineers are trained to solve equipment problems, not elec- trical problems. Vendors may be able to provide you some limited assistance in helping determine if you have a problem, but the final selection of the proper equipment, and ensuring that it has been properly installed is ultimately yours. It's your installation.

Knowing what type of conditioning devices can provide the proper, most cost-effective protection for your site can possibly prevent an expensive mistake, and help ensure a smooth DP operation.

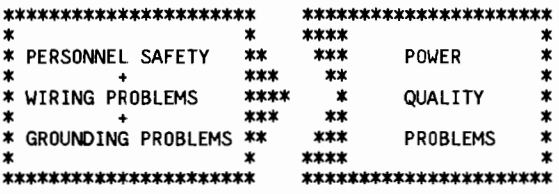
POWER PROTECTION EQUIPMENT

The first thing to do before any power protection equipment is purchased is DETERMINE WHAT THE PROBLEM IS. Power problems are solved by addressing two separate but related areas:

(1) Correct wiring configuration and grounding, and

(2) True power problems (transients, sags/surges, blackouts, and frequency shifts).

Wiring and grounding MUST be correct before anything will work properly. It is generally the case that poor equipment performance is a result of poor wiring/gounding and/or true power problems. System reliability cannot be achieved without addressing both concerns.



* * * * *

ISOLATION TRANSFORMERS

As a general rule, no stand-alone isolation transformer, when actually installed, will meet its published isolation specs. This is because of the many installation variables and non-standard wiring practices which cannot be controlled by the transformer manufacturer. Isolation transformers can be installed improperly, causing unknown safety problems or continued poor system performance.

Proper connection and grounding of the transformer is more important than a high db rating. An improperly installed isolation transformer may in- crease, or do nothing to alleviate, common mode noise problems.

1. Isolation transformers will usually resolve transient problems if they are properly installed. They cannot resolve voltage fluctua- tion problems.

2. A standard power transformer will transfer both power and "noise" from the primary to the secondary. Because the windings of a power transformer cannot respond to high frequences, transients will easily pass from the primary winding to the secondary (output) side.

3. If the transformer manufacturer installs a "Faraday Shield" in the transformer (consisting of a grounded, single sheet of conductive copper foil, spanning the full width of the primary and secondary windings), this shield will shunt the majority of the noise impulses to ground. This type of transformer is also called a single- shielded, or elecrostatic transformer. Electrostatic transformers will generally reduce common mode noise by a factor of 1000:1 (-60db), or greater.

4. Two additional steps can be taken to produce ultra- or super- isolation transformers. A "box shield" can enclose the primary and secondary windings. This shield provides a second path for shunting noise on the primary (input).

5. The next step in the "ultra-isolation" process is to wrap each indi- vidual coil in another layer of conductive shield and enclose the entire transformer assembly in a shielded enclosure. These expen- sive, very high isolation units are also known as triple-shielded transformers. Their common mode noise rejection (UNINSTALLED) can be greater than 10,000,000:1 (-146db). It is HIGHLY unlikely that these figures will ever be met (or even come close) in an actual installation, because of the many installation variables. These numbers are test lab figures.

6. Before you attempt to solve a "noise" problem, make sure that you have accurately identified the problem that really needs solving. Most computer power supplies can effectively filter normal power- line noise.

7. The logic for solving noise problems with an isolation transformer is exactly opposite that for making an audio frequency transformer work. In the isolation transformer, the 60 cycle hum is the desir- able component, and other frequencies are undesirable.

8. An isolation transformer MUST be installed safely and properly. Manufacturer's instrucions must be followed precisely, or noise may bypass the transformer and go directly into the load. The neutral on the secondary must always be grounded. "Floating" the neutral is a code violation, and can cause many problems with switching type power supplies. If you (or your CE) measure greater than 1 VAC

between neutral and ground, you are facing a situation that requires immediate attention.

POWER LINE CONDITIONERS

A powerline conditioner is a device that reduces noise and compensates for voltage fluctuations. There are two primary types of devices on the market that will do this. Each type of unit has its advantages and disadvantages.

One type of powerline conditioner uses ferro-resonant technology. Ferro- resonant units are very reliable, and provide smooth voltage regulation. This type of unit is usually initially less expensive to purchase than tap-switching powerline conditioners.

Ferro-resonant line conditioners have some inherent deficiencies which should be carefully considered. Ferro devices work best when they are heavily loaded (70-90%). This may leave little room for future growth. A lightly loaded ferro-resonant regulator's voltage output may be higher than 120 volts. The voltage can go as high as 126 volts. These types of devices need heavy loading to saturate the transformer and obtain the ferro resonant/voltage regulating effect.

Perhaps the biggest problem with ferro-resonant devices is that the output voltage may collapse if the power units are approximately 150% overloaded. This can happen when discs are spun up, or line printers powered on. Some sites have had to modify their operating procedures to accommodate the powerline conditioner. To alleviate this problem, some manufacturers sug- gest the installation of two units--one for the CPU and peripheral, and one for the discs. This can increase installation costs significantly.

The second type of line conditioner on the market today is an SCR (Silicon Controlled Rectifier--a type of elecronic switch) tap-switching isolation transformer. This type of unit uses SCR's to electrically change the length of an isolation transformer in response to voltage variations. This stepped up/down voltage is then filtered to reduce powerline noise. Tap- switchers are usually initially more expensive than ferro-resonant devices.

Voltage regulation is performed by electronically switching the transformer winding taps on and off. Tap-switchers are not sensitive to loading. The output voltage will vary in step functions.

A bypass switch is a must to ensure continued operation should the voltage regulator fail.

Tap-switcher output voltages are not affected by high overloads. One properly sized unit can usually power the entire system.

Tap switchers cost more than ferros. However, this initial cost is usually recaptured in less than one year by power savings. A lightly loaded ferro will be about 65% efficient. A tap-switcher is about 95% efficient. Considering a 15 KVA load, this difference in efficiency is about 3 KVA per hour. With power costing about $.06 per kilowatt hour, this savings could be as much as $5 per day. In larger installations, the ferro cannot be put in the computer room because of the heat and noise. Tap switchers are quiet and run much cooler.

*****

Assume a 50% load (7.5 KVA fed from a 15 KVA conditioner):

If 7.5 KVA (out) = 95% efficiency (tap switcher), then 100% (input power required) = 7.89 KVA power required to operate the conditioner and the system.

If we have 7.5 KVA (out) at 65% efficiency (using a ferro device), then 100% input power required is 11.54 KVA.

```
 11.54 KVA
- 7.89 KVA
----------
  3.64 KVA/hour difference X $.06 per KWH = approx. $.21 per hour
       X 24 hours = approx. $5 per day X 365 = approx. $1825 in
       additional energy expense.
```

*****

All power conditioning devices must be properly installed. The power input and output leads should be kept separate. There must be a single- point, electrical (not mechanical) ground. Only the critical load should be connected to the power conditioner.

The load on the conditioner should be balanced.

COMPUTER POWER CENTERS

Computer power centers offer many advantages. They are most cost effective with multiple System III/44 or 64 installations. They are a good way to completely eliminate workmanship variables and power system design inter- pretations. They give the DP manager control and monitoring capabilities over the power system. They offer a standardized integration scheme for incorporating safety items. Relocation and upgrades become much easier. Power centers provide the end user with a standard engineered power system which effectively takes care of the majority of power problems--wiring, grounding and transients.

If available, use a dedicated 480 volt feeder to supply the power center. 480 volt services are relatively "stiff" and have a tendency to resist voltage sags. System installation costs can be reduced because the wire and circuit required for 480 volt services is about half the size of those required for 208 volt installations. These feeders should ideally originate at the building service entry. It may be possible to eliminate an additional voltage regulator requirement if 480 volt lines are used.

The powerline feeder will terminate in a J-Box, supplied by the manufac- turer. The power center is then plugged into the J-Box. The power center should contain a copper core, electrostatically shielded transformer (to remove transients), power distribution panels and flexible, shielded, waterproof cables with the connectors attached, and power/voltage/monitor- ing capabilities. Any device in the computer room can be easily relocated without an electrician. New circuits can be added as desired. Safety equipment (halon, emergency power off, etc.) can be integrated in the power system.

All power centers, although they may appear on the surface to do the same thing, are, in fact different. Many engineering differences exist between different vendors which cannot be determined from sales literature alone. Examine various units before making your final decision.

Ensure that your unit can be easily serviced without system shutdown if the power center elecronics should fail. Ensure that the power center has a copper, not aluminum, transformer. Are service contracts available? Is there sufficient room for additional future cables? Who will install addi- tional breakers and cables as your system grows? Will it be necessary to power down to add new circuits (this sometimes has to be done when bolt-in breakers are used)?

Is the unit efficient? Will it add significant heat load to the computer room? Look inside the unit. Are the input and output lines separated to minimize noise transfer? Does the unit construction seem to be of high quality? Does installation and integration of the power conditioner into your existing alarm and sensor system seem as if it will be a straight- forward project?

Computer power centers are an excellent complement (not replacement) for motor-generators and UPS. Some manufacturers now offer units with voltage regulation. A bypass is a must for continued operation should the voltage regulator fail.

* * * * *

MOTOR GENERATOR SETS

A motor generator (M/G set) consists of an A/C motor which is mechanically coupled to a generator and integrated with the appropriate control cir- cuitry.

M/G sets are very resistant to transients, brownouts, and voltage surges. They contain sufficient mechanical (rotating) energy to "bridge" short-term power outages. An M/G is not an Uninterruptable Power System.

When the input power stops, the output voltage will continue to stay up for a short period (typically, 1/4 to 1/2 of a second).

Most power interruptions in the Los Angeles area are short-term breaks-- light flashes. If your requirement is the ability to "ride-through" a short-term power break, and also desire power protection and voltage regulation, an M/G set can be a good selection.

Reliability is important. A properly designd M/G power system should have a "bypass", so that if the M/G set fails, the entire computer system will not have to wait until it is fixed.

M/G sets usually cost about 1/3 to 1/2 of a UPS. M/G sets also have the advantage of slowly "winding down", and preventing the large transient and power surges experienced during power breaks from reaching the critical load.

M/G sets have some disadvantages. The larger units are very loud, and are typically installed in separate rooms. Doors and entryways should be carefully measured before the equipment ships. Floor loading must be checked to ensure that the floor (or roof) can support the weight of the equipment. Bearings may need to be greased and occasionally changed.

M/G sets can be a good solution for customers who need power bridging and have the space and money. They are not a UPS, but they do provide many of the advantages of a UPS. M/G sets are about 80-85% efficient--comparable to most Uninterruptable Power Systems.

M/G sets can be provided with diesel generators to provide power during extended blackouts.

UNINTERRUPTABLE POWER SYSTEMS (UPS)

A UPS provides uninterrupted power for a limited time, after a power interruption.

When the incoming AC power utility fails, the storage battery system pro- vides a source of DC power to the inverter, which in turn provides a source of no-break AC power to the critical load. This "incoming power off" time is limited by the capacity of the storage batteries. Five to fifteen minutes of standby power is usually provided. This will generally provide the user with sufficient time to start an engine generator or other alter- native input AC source. Fifteen minutes is typically the time it takes for the temperature in the computer room to rise to a level where continued operations will be impossible.

After the normal utility power returns, the critical load will contunue to see no-break power and the batteries will be recharged.

UPS is complex and expensive. Large systems usually require the assistance of a consulting engineer specializing in UPS.

*****

Article 645-3 of the National Fire Protection Code requires specifically for Data Processing Rooms, that a disconnecting means be provided as part of the main service wiring which can be controlled from locations readily accessible to the operators and at the principle exit doors. This "shunt- trip", or Emergency Power Off, or Contactor, when activated, disconnects power to all electronic equipment in the computer area and shuts down the air conditioning. The room lights should remain on.

All computer rooms should have an Emergency Power Off as part of their disaster planning effort.

STATIC

All digital circuits can fail as a result of static discharge. Although the actual currents involved with static discharges are small, the dis- charge voltages developed can easily reach 20,000 volts or more. There are primarily two things which can be done when dealing with static problems. You can attempt to make the device more tolerant of high voltage dis- charges or you can attempt to prevent high voltage discharges from occurring.

Line printers generate static as the hammers strike the paper. As the hammers hit the paper thousands of times, high charges are developed. In low humidity areas, it is possible to generate 500 volts per printed page. Putting 20,000 volts+ on a stack of paper is not uncommon. When that voltage discharges, the results are unpredictable.

Static control can take many forms, but they all have one thing in common--providing some type of conductive pathway to ground is necessary to prevent high voltage from accumulating. Devices commonly used for static control are antistatic sprays, conductive mats and rugs, humidifiers, and tinsel and corona discharge devices. Good grounds are a must for static suppression devices to work effectively.

All these devices work, but some work better than others. Line printer tinsel strips are very common. There is nothing to break, and they are easily checked by visual inspection. A tinsel strip will remove about 90% of the static charge on a sheet of paper. The charges

continue to accumulate--they just take longer to manifest themselves. Using a grounded paper stacker with tinsel generally does an adequate job.

Proper power grounding plays an important part in eliminating static. Poor grounding compromises the reference needed by the peripheral. If the static voltage discharges through a poor ground, noise may be generated, which may travel directly to the CPU via the peripheral cable through a CRT. This may directly affect system operations.

BIO SKETCH - ED MUXO

1980 - Present: Founded Computer Power Solutions, Inc., specializing in computer power consulting services and power conditioning equipment sales.

Consultant to several mainframe manufacturers. Published articles in Computer Decisions, Datamation, Hardcopy and System 34 World magazines. Listed in California Who's Who. Member DPMA and DECUS.

1972-1980: Employed by Hewlet-Packard in field service and management positions. Installation planning and power specialist for Los Angeles, Fullerton and San Diego offices.

1969-1972: Field Service Engineer with Burroughs Corporation. Respons- ible for Bank of America data processing center.

1962-1969: U.S. Navy Polaris missile computer and instrumentation specialist.

Software as a Long Term Investment

by Birket Foster
M.B. Foster Associates Limited

These are the main sections of this paper, and form the basis of a framework for planned acquisition or creation of software to support a business. The role of the information systems department is a support role. The objective of the information systems department should be to deliver the software which supports the business decisions made by the organization. The key is to do this in the most cost effective way possible just as any other section of the business would perform their role. By ignoring the "Black Magic" aspects, which used to be associated with the data processing profession, and concentrating on treating software as a long term investment, better business decisions will be made. The software investment is to be made in the same way as justifying a new piece of equipment. Based on the long range business plan the costs and benefits, as well as the return on investment is to be calculated for each software project. Since the plan is long range the investment can take into account the long range information needs and grow with the company.

1. Know Your Business

The history of data processing is full of unsuccessful projects, continuing projects and ever-changing projects. If many of these situations were examined it would be found that the data processing department was not aware of the direction the organization was going in, nor of the information needs of the users of the system.

In order to treat software as an investment you have to examine the corporate business plan. Software is supposed to support the business operation. To be able to support the business the software must be there ready to run when it is required. The longer time frame that the business plan covers, the more likely that the long range information needs can be taken into account.

If the business plan calls for doubling the volume of sales over the next two years, the software should be capable of handling the increased volume of transactions necessary to support it. The solution in some cases is to buy a larger machine, more terminals, more disk or more memory. The hardware expansion should be phased in in increments appropriate to the business's growth. Even with the expansion of the hardware, there might be software limitations. The modifications to the software to adapt to the increased volume might not be necessary if the software is designed from the start with an eye on expansion.

The information systems department (ISD) must be included in the corporate planning sessions. The corporations which intend to survive in competitive marketplaces have to be able to use the data they can collect in innovative ways to make better decisions. By including the ISD in the planning sessions, the systems to deliver this information can be indentified, designed and implemented with the overall corporate plans and objectives in mind.

2. Chart Your Systems

As part of the planning exercise the organization can identify the systems required to run the business. The systems fall into the three categories identified by Peter F. Drucker - Operational, Managerial, and Strategic information systems. The operational systems are the ones which do the tasks required for day-to-day operations such as purchasing, invoicing, order entry etc. These systems will exist to a greater or lesser exent in any organization. The systems may well even be manual systems or a combination of manual and automated.

The next level of system is managerial which is usually a summary of the information captured at the operational level. The management decisions are made based on this information. The key to getting this information collected is to know the decisions that the management will be making and ensuring that the data is captured. By examining the corporate wide information needs the ISD will be able to determine the best way to organize the data to support both the operational and managerial decisions.

Certain of this level of information may come from outside corporate boundaries (for example Credit Rating information) and must be integrated into a corporate data plan. Each of the various systems must be identified and then the major functions identified. A matrix type arrangement for identifying the users of the data, the providers of the data can be quite helpful. Each system description should include the manual as well as automated flows, the sources and destinations of any data, including the flow into other systems. The volumes of data, the timing and the expected growth over time should also be included.

The highest level of information is that which supports the strategic planning. Most of this information will come from outside the corporation and will be married with highly summarized information (trends) from within the corporation to assist in the long range strategies. This level of information is also used in the goal setting process.

The key to developing the systems to support all these levels of data is to design the data structures carefully. There are several authors who have written about how to implement the structures. One of the books which we have found most useful is "Strategic Data Planning Methodologies" by James Martin. The basic fundamentals behind this book are easy to follow and the book can be used by both Data Processing departments and User management. The matrices which are developed in the course of the analysis can be used to assist in the maintenance of the system as well as to assist in the documentation of the system.

The step-by-step approach developed by Michel Kohon (Supergroup july/aug) will assist in using the matrices in the most effective manner.

3. Select Your Priorities

Knowing where the current systems stand is the first part of the battle. Now the problem becomes to chart the plan for getting the information systems in place to be able to support the business plan. As with any other business decisions the problem of limited resources will apply. There will be choices to be made in terms of allocation of funds, use of manpower, availibility of equipment and all of this will be constrained by time factors. From the list of necessary systems a priority should be established. This is a business decision and should be made to get the best return for the dollar.

When selecting the priorities choose your options carefully.

-Remember to weigh the risk involved in developing the system.

If the system is critical to the business remember to allow for the resources to ensure successful completion. The users are considered to be a critical resource in all systems development projects and in software selection and package implementation. By narrowing the scope of a project an incremental approach to implementation can be taken. Since the plan will have identified all the necessary functions and data this approach will allow a creeping commitment with results coming earlier as sections of the system are completed. When accounting for the risk include user time, training time, testing time and all other costs associated with the implementation. If this is not done the cost of the project, and the cost/benefit analysis as well as the true investment at stake will not be properly defined.

-What would be the time to redo the system if this was required?

It is possible that even with prototyping the system originally envisioned will not be the system required. If this is the case the incremental approach will allow the minimum investment of resources and the maximum flexibility in development options.

-What would be the effect on normal business operations if the system is late being delivered or fails completely?

Just in case it would be helpful to have a contingency plan ready and to know what job functions will be affected. What system will be used if the scheduled implementation dates slip? Given expected volume, computer resources available and manpower, would the

normal business operation be able to continue with business as usual? How long could this be kept up at what cost?

-What is the upper management commitment to this system. The commitment should flow from the long range planning excercise and must be there if the system is to be a success.

If upper management is not convinced of the importance of a system, the resources required to complete its implementation may be hard to acquire. To deliver a system aimed at other than upper management is to invite trouble. The management of an organization must determine the entire nature of the business including the priorities for the Information Systems Department. Since the ISD is a support organization let the management call the shots on the priorities and choose what should get done when.

-How does the system effect the overall plan? Are there other activities which depend upon this systems completion on-time?

The more systems are implemented in parallel or in small segments serially, the more important project management will be. The importance of any system to the rest of the information system plan must be recognized and the appropriate resources allocated. The higher the risk the more precautions to ensure success should be taken. By recognizing the potential risks, the contingency plans can be put in place to assist the implementation schedule.

-Have the long term integration plans been taken into account? Have all the bridges been identified?

Most systems are not standalone in a corporate information plan. Be sure that the analysis has identified all the interfaces. If the system is evolving, be sure that the "bridges" to get from one stage to the next have been identified. All implementations should be thoroughly planned to ensure that normal business operations continue with minimal disruptions. Each stage of implementation should take into account the future of the system so that redoing or retrofitting is minimized.

When selecting the priorities remember that the objective is to get the best return on investment in the shortest possible timeframe.

4. Choose Your Software

The software can either be made or bought. Today it is usually more effective to buy packaged software than to attempt to create systems from scratch. Most applications have already been created for the HP-3000 and can be purchased and maintained for a fraction of the cost of doing them in house. Software selection is often best when the user is the driving force. The following points should be borne in mind:

The long range plan may call for systems which have to be integrated. Make sure that the systems is capable of fitting into the plan.

Beware of multiple vendor situations especially if the systems have to be closely coupled. On the other hand a single vendor should have available all or most of the systems you require for the long range plan. Buying a fantastic GL with no interfaces to other systems may be more of a hinderance than buying a good GL with good interface definition.

All costs including hardware, conversion, software tools, training and manpower should be identified so that the estimated cost of a system can be budgeted for and the real return on investment measured. These real costs should be used as feedback into any cost estimation process. (We should learn from mistakes).

With the market trend towards micro computers comes new opportunities to allow the user better data manipulation capabilities. If the best opportunities for cost effective systems involve a combination of micros and minis then the software will have to fit several different criteria. Software projects on a micro are quite often completed by using the available packages or by using fourth generation tools. The cost invested in a micro based project can be considerably less than on a mini, but the user must be made aware of the risks and must be willing to take on tasks such as backing up data, which were previously the duty of the data processing department.

The most important trend in the computer technology today is the micro-mainframe interface. This calls for any application which is being designed to take into account the needs of all levels of users. The reports at the senior management level require highly summarized data with short retrieval times. The databases designed to accomodate this level of reporting will have to allow for summarized information. The reason for the popularity of such packages as Lotus 1-2-3 or the other Visi-clones is that the results of what-if analysis are available almost instantaneously. The new data structures will allow for the transfer of the information needed for these kinds of analysis with the minimum of difficulty. Choosing sofware combinations which allow for this activity will become increasingly important.

Other trends which will affect the way software is chosen are micro-networking and Unix like operating systems. The combination of

these trends leads to some very powerful configurations. The rumour that HP may have developed a 48 or 64 bit operating system should clue those looking for long-term software investment to be aware of what is the transportability of their systems. The use of languages such as "C" or Pascal may come into vogue to ensure this compatibility. Using 4th generation languages to create programs in these languages will become more important. Even COGNOS (formerly Quasar Systems) is beginning development in "C" with an eye to the future. As processors become more powerful and memory and disk storage less expensive it makes more sense to aim for systems which are easy to maintain and quick to build. Prototyping is the way of the present, the way of the future is for the user's to assist in customizing the templates for applications provided by the software tool suppliers.

Networking micros will free the mainframe (mini) from many tasks and its role will become that of a database machine. In this role it will provide data to the user's and handle the number-crunching power required for such activities as month-end invoicing etc.. By 1989 over 25% of the micros are expected to be portable. These are already showing up with portable hard disk (the Eagle Spirit and the Compaq portable with 5MB) are just part of the new generation of micros. (It is interesting to note that both Bytec makers of the Hyperion micro and Compaq are using their in house HP 3000's in conjuction with the micros to help run their businesses.)

The software development will be done with an eye on the long range corporate plan, using packaged software and 4th generation development tools (sometimes called productivity tools). The objective is to make the investment in software easy to maintain and flexible to the changing corporate requirements.

5. Review Your Software Implementation Plan

Your plan is your guide but must be reviewed from time to time to make sure that changing circumstances have not changed the corporate priorities. By implementing on an incremental basis the risk is minimized. From time to time systems which are underway will have to be scrapped. The fact that systems are prototyped will speed the implementation process but will not change the fact that the system development team must be aware of the business and its long term goals and objectives. A continuing string of prototypes or partially developed systems won't buy success. Do it right the first time is still the motto even in todays world of application development.

If the need for a system changes the development group must be made aware of the new priority as soon as possible. A plan is not cast in concrete and should be adjusted as needed. The development should be user driven according to the business needs of the organization, in accordance with the long term goals and objectives. These needs should be addressed as a business decision in each case and changed as a better return on investment of resources becomes evident. By using the step-by-step method or similar results oriented, short run project philosophy the risks will be minimized.

The overall plan should be reviewed regularly and changed accordingly. The investment in information systems must be considered carefully and forced to deliver quantifiable results. The investment should be made cautiously remembering that as the business changes so will the information requirements. The basic (general) information requirements should be satisfied first as they are unlikely to radically change over the course of the years.

6. Conclusions

The recent recession has caused many organizations to rethink their entire way of doing business. The value of accurate and timely information has been recognized. The cost of delivering this information is becoming a major part of organizational budgets. The short run fix is too costly a solution as many organizations have spent too much time developing and redeveloping the same system. The cost of planning a long range information strategy is much less than the cost of maintaining a continuous development environment. The cost of implementing a system includes the user involvement, the systems department involvement as well as the the hardware cost. More often the investment in planning will be placed in corporate budgets and fully costed.

The "Small World" will have its impact on the way organizations operate. For this reason software which can interface micros with mainframes (minis) will become a necessity. In some cases it will be possible to run a software package on both the micro and the mini. In most cases the transfer of data will be most important.

The use of database machines (such as the one Eugene Volokh is working on) will become more and more important. Accompanying this trend will be the use of more powerful software to be able to retreive data in the format required quickly. DISC's IMSAM product allows very rapid retreival of data using a variety of methods. This speed is due to the use of a B-tree structure, something which has been showing up in most relational databases.

The long term will be the framework for each implementation which takes place. Whether

the software is made or bought, the user will be more involved than ever and upper management will want the best bang for the buck. The information service centre of a corporation is going to have to compete for the organizations budget dollars just as every other department. Necessity is the mother of invention, so you can bet there will be a great deal of imagination applied to how to extract the most out of every bit of data available to the organization. The systems department will work a lot more closely with the users. Its a small world it just goes a lot quicker, but it's evolving as it should.

Birket Foster is based out of Ottawa. He was instrumental in developing and implementing the initial marketing strategy for Quasar's QUIZ product. He is chairman of SIG-SOFTVEND the special interest group for software vendors. His company provides marketing planning and support to software vendors and software selection services to users. THE CATALOGUE, a comprehensive publication describing the third party HP3000 software is published by M.B. Foster Associates Limited. This paper was word processed using a HYPERION Personal Computer which uploaded it to an HP3000 using its built in software and modem. The HYPERION was designed and built in Ottawa to be IBM-PC compatible by the BYTEC group.

Accounting Rods: New Visibility with Analog Financial Statements by DSG/3000

Sam Boles

Hewlett-Packard

Synopsis

In recent years the investment community, the government and the accounting profession have focused attention on the meaningfulness and intelligibility of financial statements, to match their accuracy and timeliness.

Computers have made a major contribution to the issue of accuracy and timeliness. Computers are beginning to make a contribution to the issue of meaningfulness and intelligibility.

A team of development engineers and performance engineers at Hewlett-Packard have devised an HP3000 implementation of financial statements -- traditionally presented in digital form -- in analog form using the Vernacular of the Rod as conceived by the musician/teacher Georges Cuisenaire and adapted for accounting statements by Dr Irwin Jarett.

Cuisenaire, with the native mode of the human CPU and I/O subsystems being analog, evolved the Vernacular of the Rod to facilitate understanding the analog-to-digital conversion fundamental to mathematics, the second degree of abstraction of Aristotelian epistemology.

The application of computer graphics to produce financial statements and ratios in this vernacular provides visibility into the health, performance and trends of a business enterprise that is both efficient and effective because of its intuitive nature -- an interesting blend of vintage psychology and the leading edge of computer business graphics technology.

Speed and Comprehension Out of Synch

Faster than the Speed of Time

If you're near Heathrow about 10 o'clock in the morning, you can hear a jet roar in the distance. If you look over in the direction of the airport you can see the Concorde Supersonic Transport climb, bank to starboard, undroop its nose and disappear quickly into the West.

It will touch down at Kennedy in New York *an hour ago. Not an hour from now.* An hour ago. It will get to New York in under four hours. That's faster than the time will get there. Faster than the speed of time.

That's fast. The Concorde even *looks* fast. Even when its sitting in a hangar. Next time you're on your way to Cornwall stop by the Royal Naval Fleet Air Museum at Yeovilton in Somerset, a couple of hours west of London. The original Concorde is there where you can walk up and touch it, go inside, see the incredible array (12 tons) of instrumentation used in debugging it (3000 sensors).

(That's about 50 miles from the mysterious prehistoric edifice at Stonehenge . . .

. . . just like the British to put those two together. That must have been what they had in mind when they

invented the word "anachronism." Or maybe, if you believe some of the theories about Stonehenge, the word "convergence").

The Staging Area for the 21st Century

Anyhow, that's the way it is here in the mid-80's, the Staging Area for the 21st Century. *Fast.*

And if you think the SST's fast then *take a look at our computers.*

We've got a laser printer that prints 45 pages a minute. It's got a feature that prints these pages 4-up on a single physical page. That's the equivalent of over 10,000 lines a minute on a impact printer. *Fast.* (Actually, we position it as a mid-range performance printer - so it's *not* the fastest in the world).

If the average annual statement of an average sized company is 20 pages, you could print *2 of them every minute*, or 8 of them a minute if you stacked them 4-up.

But Who Can Read That Fast?

If you take your top speed-reader accountant or investment analyst or acquisition lawyer or stockholder and give her two annual statements a minute to read, you have a problem.

If the Concorde gives you jet lag, you might call this computer lag or maybe computer (user) lag.

Our computers are so fast they can give us *more information in a shorter time than our physical and mental learning facilities can handle.*

Even If We Could, What Does It Mean?

We've come a long way from the Era of the Green Eye Shade. Computers have eased the crisis that we used to have at the end of every accounting period when we'd try to close. It took an army of bookkeepers working through the night, trying to find that penny off, wading through the lower level constructs, building up to the final statements.

Computers have brought relief to the speed and accuracy issues of closing and producing financial statements. So much relief, in fact, that folks interested in the accounting profession -- like the American Institute of Certified Public Accountants, the Securities and Exchange Commission of the Federal Government, stockholders, lending institutions and other creditors -- have started to focus on making the statements more meaningful and intelligible.

The goal is to take an annual report that is timely and accurate and gives full disclosure, and get it into a form that the ordinary prudent person can look at and get a correct message from without being an attorney or professional accountant and without spending an excessive portion of her lifetime mulling the numbers.

Some Thought on Thought

Our Native Mode: Analog

A couple of thousand years ago, in Greece, a fellow named Aristotle started thinking about thinking. How do we get to know things?

Now later such thoughts on thinking got compiled into a science they call Epistemology. As time passed, some interesting components evolved, like:

Actually, it's not thinking but dreaming.

Actually, you can't know anything for sure -- except that you can't know anything for sure.

Actually, you can't know anything for sure -- including that you can't know anything for sure.

I must think so I can infer that I exist so I can publish my book on Ontology.

Etc.

Unfeathered Risible Biped With Opposing Thumb

But Aristotle came too early to get into all this (except maybe the dreaming part that Plato seems to have told him about) so *he thought he thought.* And the way he thought he thought went something like this:

All knowledge comes thru the senses. We see, feel, smell, hear "things." As "objective reality" [whatever that may be, if anything] impinges on our sense perceptors, we capture and internalize "objective reality" in several "degrees of abstraction." Let's look at some of these:

The first degree of abstraction from "reality" is an image, a picture: I close my eyes and think "person" and I "see" this figure with head and arms and legs and all that.

The second degree of abstraction is mathematical: I think about that "person"

and if I strip away color, shape, arms, legs and other image components, I end up with an attribute we call "quantity." I'm aware of one-ness (or two-ness if I "image" two people or "n-ness" if I "image" n people).

There's also a third degree of abstraction that deals with "person-ness" or "substance" and gets into metaphysical gropings like "unfeathered risible biped with an opposing thumb" and other feeble generics -- but that can get to be quite a wicket, so we'd better stick with the first two for getting at accounting reports.

A Picture's Worth a Thousand . . .

Now those of us who think a picture's worth a thousand words probably have a little Aristotle in us. (The "picture" maps to the first degree of abstraction.)

And if a picture's worth a thousand words, it's probably *worth at least a few dozen numbers* in your typical annual report.

But let's not jump ahead without looking at an important link in the chain to analog financials.

From Greece to Belgium

Let's leave Aristotle and move a few hundred miles northwest and a millennium or two later.

There was a musician in Belgium named Georges Cuisenaire (1891-1976) who was a school teacher. One day he found one of the young children in his class crying over his math book. He just *couldn't learn it.* This touched Cuisenaire so that night he brainstormed the problem in search of a better way to teach math concepts.

He tried cutting some wooden rulers into a variety of proportional pieces. He color-coded the rods and from his musical background developed the "keyboard for mathematics."

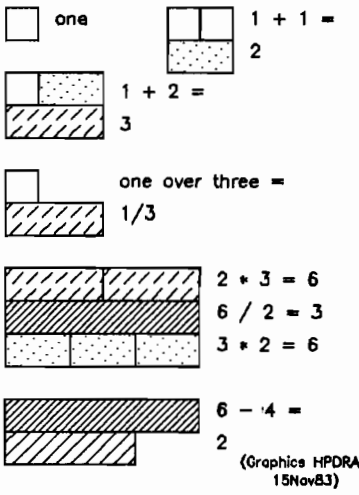
The Vernacular of the Rod

He took the rods to school with him and started experimenting. With the "little colored sticks," or "rainbow rods" as they were later called [4], he found the children were "learning by doing, learning by experience," getting to the second degree of abstraction (the quantitative) with very powerful analog images in the first degree of abstraction. The students moved out of a passive role into an active role, with their sense of sight and their tactile sense busy in the process -- one of the basic techniques of modern educational psychology.

When 5-year-old's were able to do matrix algebra through the instrumentality of Cuisenaire rods, the methodology had established that a natural gateway to the digital world of the second degree of abstraction was an analog world somewhere this side of the first (specific image) degree of abstraction -- maybe about a 1.5 degree of abstraction (generic image).

The children could intuitively "see" and "feel" greater-than, less-than and sums. A rod placed on top of another left a void at the end -- the remainder of subtraction. With a natural extensibility there came factors, commutation, fractions and base-10 concepts. The analog of the proportional rod seemed to be a natural facilitator for getting at the digital concepts. They were not memorizing formulas. They were *perceiving relationships.*

Here are some of the basics of the Cuisenaire® Rod Methodology:



The youngsters were able to "see" addition, subtraction, multiplication, division, factors, fractions, commutation, permutation. They were *perceiving relationships.*

®Cuisenaire Company of America, Inc, 12 Church Street, Box D , New Rochelle, NY 10805.

Getting It All Together for Financial Statements

. . . And Then on to the Land of Lincoln . . . .

In Springfield, Illinois, Dr. Irwin Jarett, a CPA, business graphics expert and accomplished lecturer, was investigating the problem of meaningful and intelligible financial statements. As a CPA, Dr. Jarett was not only knowledgeable of the technical derivation and implications of conventional financial statements but also sensitized to the importance of making them a vehicle of effective communication for their expanding audience.

With this background, Dr. Jarett brought together his extensive knowledge of business graphics, the current computer technology supportive of this discipline, and the Vernacular of the Rod that Cuisenaire had used so effectively in the disciplines of music and mathematics.

The resulting synthesis is a GMIS (Graphics Management Information System) that makes a positive contribution to the goal of meaningful and intelligible financial statements [3].

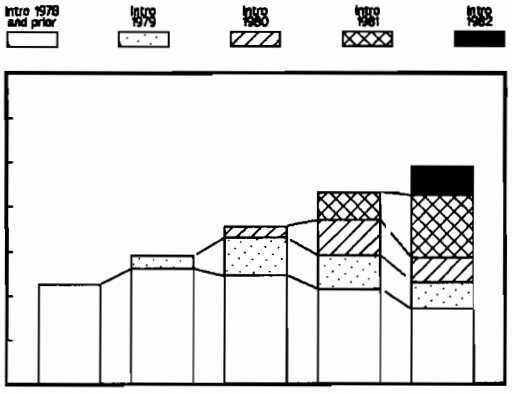
In addition to the concepts of communicating quantitative information reinforced with analog imagery, Dr Jarett's GMIS offers a structure, a systemic approach to the capture, flow and formatting of the amorphous data into an orderly and digestible product.

A DSG/3000 Implementation

Here you'll see some of the concepts of the Jarett Method used in a DSG/3000 (Decision Support Graphics) implementation on the HP3000 computer. You'll see portions of an actual annual statement from a *real company. It's not very randomly selected* but perhaps one most of us have heard of and maybe have some interest in.

Of Rods and Laser Beams . . .

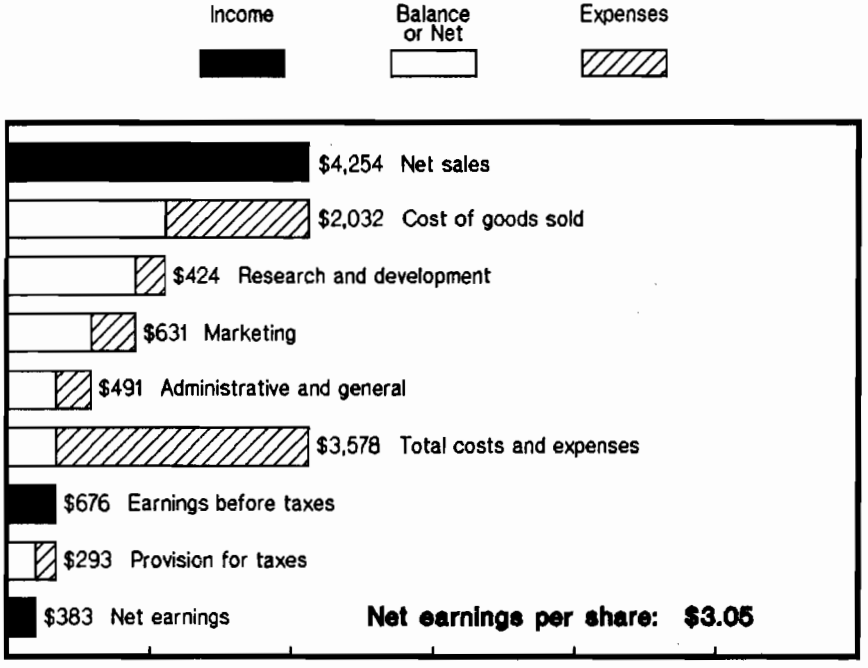
You'll see the statements presented in the Vernacular of the Rod. The final product is integrated via TDP/3000 (Text and Document Processor) and hardcopy generated through the text and graphics capabilities of the HP2680 Laser Printer to produce in a single operation the document you have in your hand. To give you a hint as to what company we've *unrandomly* selected and an example of the time-series format of the rods, here's a product aging chart:



Product Aging: 1978-1982 Orders by Introduction Group
(graphics by DSG/3000 15Nov83)

Hewlett-Packard Company

Statement of Earnings for Year Ended October 31, 1982



Millions (except Earnings per Share)

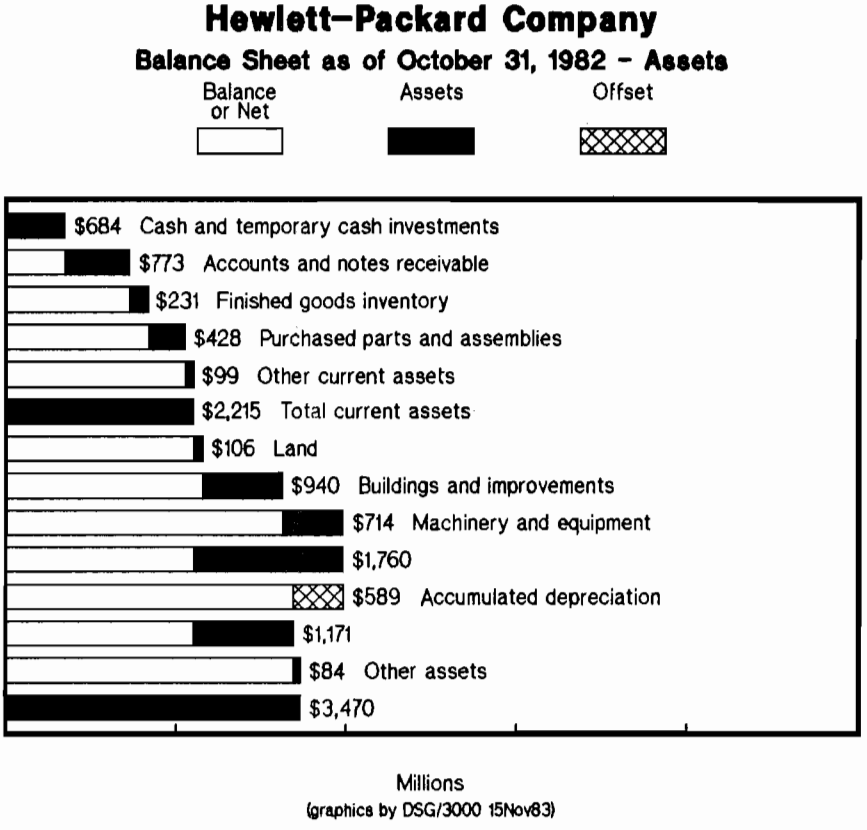
(graphics by DSG/3000 15Nov83)

The Income Statement

Here's the Income Statement with the analog rod to help give meaning to the numbers. Without looking at the numbers, you can see the big revenue rod get whittled down by a variety of the costs of doing business.

About half is for the labor and material and burden that it takes to make the product. A tenth goes into R&D as you would expect in a company whose success is based on a steady stream of innovative quality products in one of the most creativity-driven industries in human history. A sixth goes into Marketing -- getting the product into the hands of the customer and supporting it for a good return on the customer's investment. About a seventh goes into G&A, and half of what's left to taxes.

In a glance you can get a reasonably good first-order calibration on components and relativities in the Income Statement, the Anatomy of the Dynamics of a business enterprise.

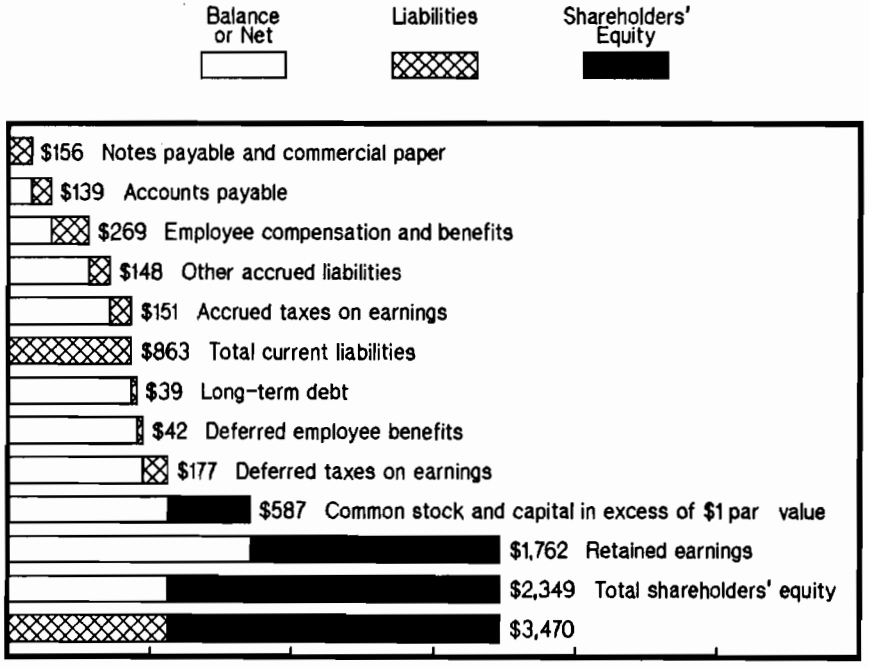


The Balance Sheet

Now for the Anatomy of the Statics of a business enterprise, let's look at the Balance Sheet. Here we can see, again at a glance, not just that it balances -- but some of the relativities of assets and liabilities, things owned versus things owed, things more current and temporary and things more long-term and permanent. We can see basic components in proportion, and groups of components, to get a feel for how it's going even before looking at the numbers.

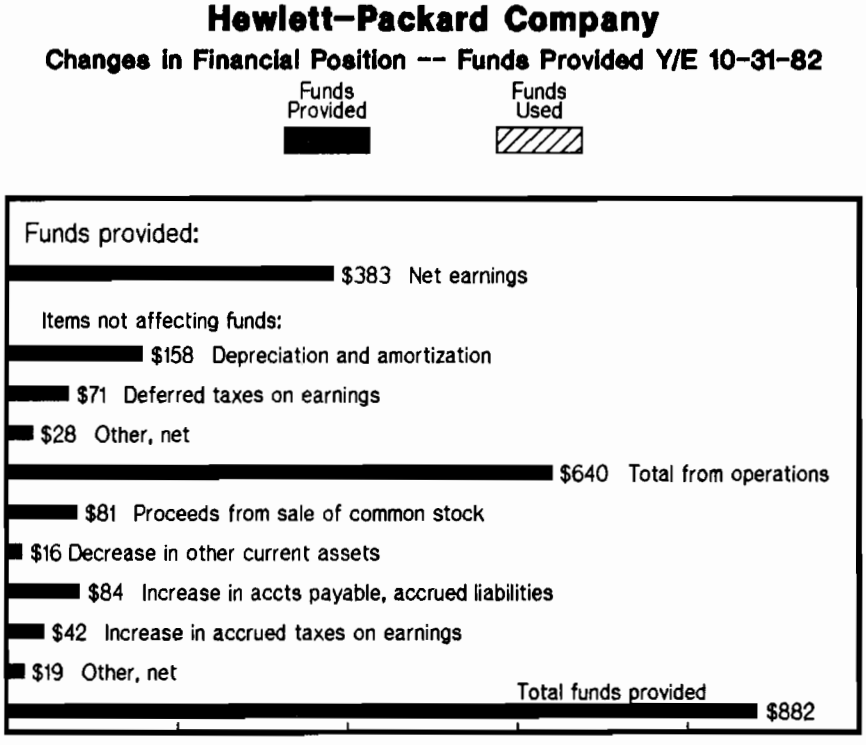
Hewlett-Packard Company

Balance Sheet as of October 31, 1982 - Equities



In the Balance Sheet we can get a message from both the big rods and the little rods. Finished goods inventory is about 1/15 of total assets, reflecting a constant pressure to keep that number no greater than is necessary to give good service to customers. About 1/5 of the total assets is in equipment, reflecting a serious commitment to productivity.

On the equity side, a noticeably small rod carries the caption "Long-term debt" (about 1% of total assets), and a big rod is captioned "Retained earnings" (about 1/2 of total assets). This reflects the conservative "HP Way" of financing an aggressive growth program (15-30% a year over the past 5 years), staying "out of the bank" and providing growth funds from earnings.

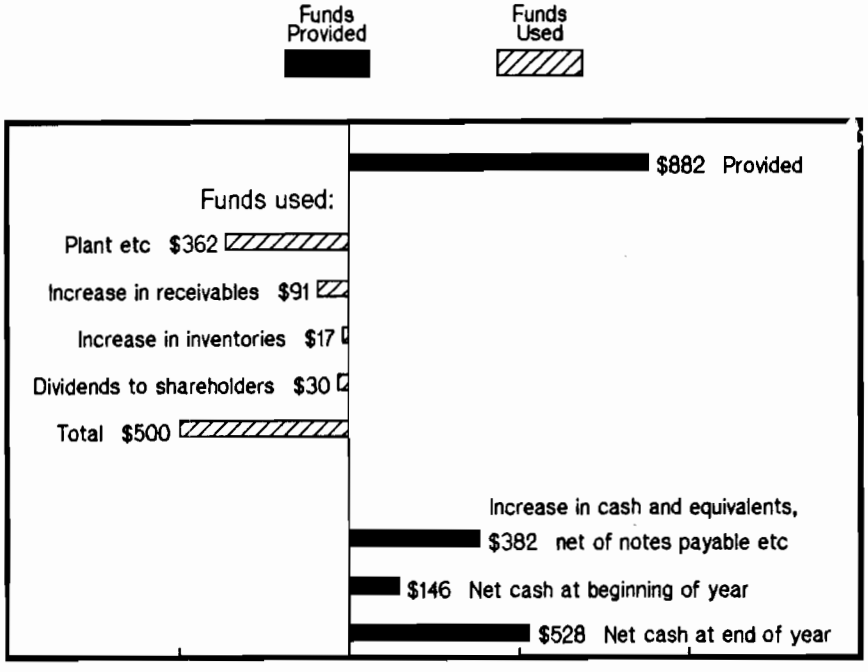


The Statement of Changes in Financial Position

After the funds-provided section there's a variation from using the rods in a components or parts-of-the-whole format. In the second section you can see variations around a zero point, with funds provided on the positive side and funds used on the negative side.

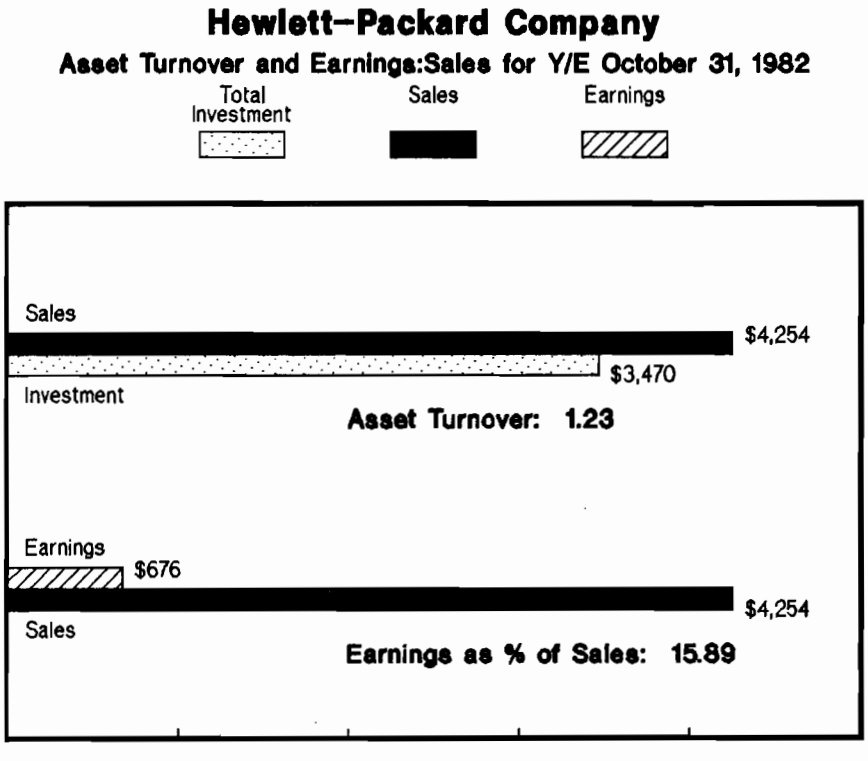
Hewlett-Packard Company

Changes in Financial Position -- Funds Used Y/E 10-31-82



(graphics by DSG/3000 15Nov83)

Some numbers the rods point up here include the big one for investment in property, plant and equipment. This indicates that the growth in sales and earnings we saw earlier is not a transient aberration but a planned phenomenon of foundation and substance. The accounts and notes receivable rod reflects some recent leasing and credit innovations to support customers in their productivity investments.



Beyond the Basics

In addition to using the rods in the basic financial statements of the annual report, the technique can give new visibility in other sectors. This is an example of fractional and division rods in action.

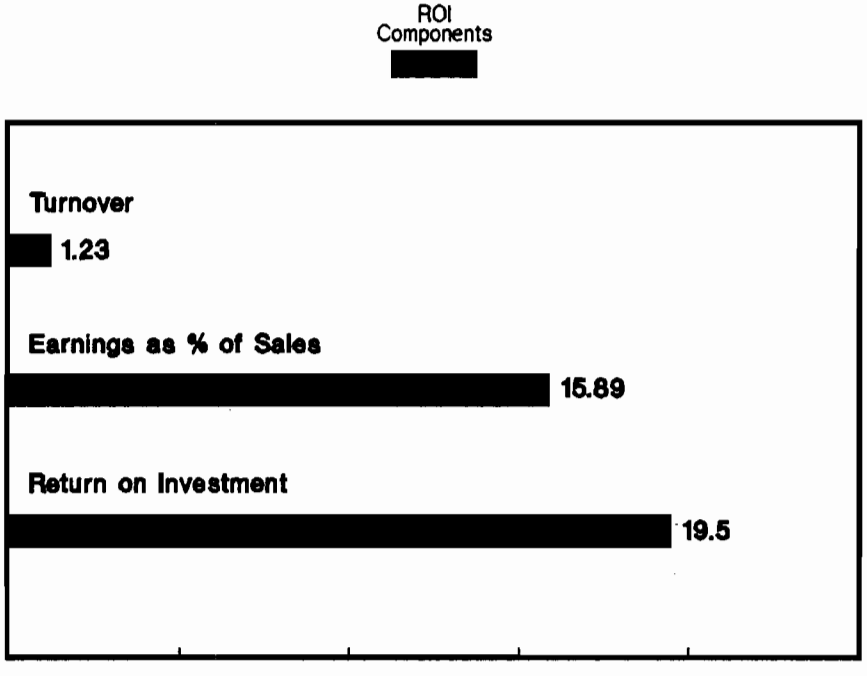
The duPont Company has traditionally rated as a well-run company. The duPont system of financial control is considered classic and is a standard component in many finance and accounting curricula in universities across the country [1].

The system focuses on a measurement called Return on Investment (ROI). A business enterprise exists primarily to make a profit. To make this profit, it uses assets. It's the profit you make on the assets you have that are the factors in ROI.

If you don't control your expenses, you cut into your margins and hurt your ROI. If you try to brute-force it with assets, your asset turnover goes down and your ROI can take a beating. If you cut into your marketing and R&D expenses for short-term expediency, you can hurt your sales and therefore your earnings and therefore your ROI.

Hewlett-Packard Company

Return on Investment for Year Ended October 31, 1982



Turnover times and percents

(graphics by DSG/3000 15Nov83)

The Anatomy of ROI

Here's a macro anatomy of ROI:

Combine your current assets with your permanent investment to get your total investment.

Divide your sales by total investment to get asset turnover.

Subtract cost of sales from sales to get earnings.

Divide earnings by sales to get earnings as a percent of sales.

Multiply this percent by the turnover rate to get Return on Investment.

You can see it here in the Vernacular of the Rod.

Conclusion

You've seen some DSG/3000 examples of the Jarett Method in action with some of the key financial statements used to show the health and trends of a business enterprise. You've seen the analog of gross income get reduced by the analog of the various sectors of expense typical of a manufacturing concern. As these various components take their toll, you are intultively aware of relative impact from the relative size of the analog.

You've seen working capital ebb and flow; you've seen the Balance Sheet do just that, with a graphic analog reinforcing the numeric representation.

The Cuisenaire® Vernacular of the Rod is a vehicle to *transform data into information.* It's intuitively meaningful. The Jarett GMIS is a structure and discipline for implementation. And DSG/3000 is a productivity tool in using the Vernacular of the Rod to give new visibility to financial statements.

Bibliography

[1] Brigham, Eugene F, and J Fred Weston, *Managerial Finance* (New York: Holt, Rinehart and Winston, 1966).

[2] Finney, H A, and Herbert E Miller, *Principles of Accounting* (Englewood Cliffs, N J: Prentice-Hall, Inc, 1965).

[3] Jarett, Irwin M, *Computer Graphics and Reporting Financial Data* (New York: John Wiley & Sons, 1983).

[4] March, Rosemary, "Georges Cuisenaire and His Rainbow Rods," *Learning* (November 1977), pp 81-86.

About the Author ....

Sam Boles *is a Systems Engineer in the Systems Performance Center at the Hewlett-Packard computer facility in Cupertino, California. With HP for seven years, Sam's computer experience started back in the AUTOCODER days of the 1401/1410, migrated thru the 360/370 era, and now focuses on HP hardware and software performance characterization. Sam earned his MS at UCLA in Information Systems.*

sebiug42 2300 15Nov83

Supporting Micro-computers - Small World but lots of consequences

by Birket Foster
M.B. Foster Associates Limited

1) Introduction and definitions
2) Hardware support
3) The microcentre
4) Software support

Introduction and definitions

The advent of the personal computer has caused several different aspects of information processing to change. The rapid change in technologies associated with the microcomputer revolutions means that the machines bought today will soon be outdated. The obselecense can be caused by outdating in either the hardware or the software. The problem is that although the investment in any one workstation is a fraction of the cost of a larger computer, micros require software, backups, documented procedures, and hardware maintenance.

The process of supporting a group of micros must start with a few assumptions.

1) Micros are to be justified over a useful life of 2-3 years.

2) Micros are to be used only in those areas which the usual data processing methods cannot respond or respond as quickly as the end user requires. Most of the applications which fall into the latter category are spreadsheets, wordprocessing, databases, access to information services, electronic mail, and decision support systems.

3) Micros are being used to support the job functions of professionals. Keeping the micro from wasting the professional's time trying to get the hardware or software to work is of prime importance.

4) There are a number of levels of support available the cost of each is varies as does the response time.

The ultimate in hardware support would be to have each workstation completely duplicated. This option however becomes very costly as the ratio of workstations to professionals becomes higher. For software support there would be a computer specialist to assist each professional in applying the hardware and software to his/her job.

The information centre concept addresses this problem by centralizing purchasing, and hardware/software support. Generally an information centre will support up to 3 brands of micros for the general user with exceptions for user's whose application software does not run off the selected brands. To be able to support the users adequately there are often a limited number of approved software packages and peripheral options (printers, modems, plotters, etc.) which the centre will provide guidance and support for.

The problem is not so much finding the correct hardware support, as realizing that the micros are going to become assistants to the professionals and will by the late 1980's be as commonplace on management desks as telephones. The problem to be addressed is simplified when viewed from an economic standpoint.

> If the life of a micro is 3 years, the number of hours of work per year is 2000 then the cost of the micro is divided by 6000 hours. Even if we include the cost of maintenance, software, a plotter and a printer to raise the cost to $12,000 per work station then the cost is only 2.00 per hour. The cost of the micro per hour

is much less than the cost of the professional. If a loaner micro was available for every 10 micros in use the cost would only cost $2.20 per hour/station for the 10 stations. This also takes the immediate time constraint off the repair cycle and allows less costly modes of repair to be contracted for. The problem therefore is to ensure the machine and software are always available to the professionals who are using them. This involves both hardware software and application support.

Hardware support

There are several aspects to hardware maintenance. The cost of hardware maintenance varies for the micro just as it would for the mini computers or mainframes. The cost is a factor of the level of service desired. The level of service should be driven by the requirement to support the professionals using the machines with the minimum of disruption of their normal work. Remember the machine is only a tool to help the professional to do his job. You wouldn't keep a car which broke down all the time nor one which was hard to control.

Costs associated with hardware maintenance are:

Labour
Parts
Tools
Travel (if onsite)
Shipping (either parts or machine)
Loaner equipment
Training

When hardware is chosen there should be local support if possible for it. The cheapest support is actually to mail the piece of equipment back to the manufacturer or dealer. High 0;1mMmean 0;1mTmime 0;1mBm etween 0;1mFmailure (MTBF) statistics often justify the risk of allowing equipment to be placed in locations which do not have service facilities nearby. District offices can take advantage of the new technology knowing that if there is a breakdown of a machine a replacement can be swapped in from the region within 24 hours.

What are the ulimate options on hardware maintenance?

Full onsite spares, loaners and maintenance personnel:

Advantage: Quick repair of machinery, if many machines can assure all have preventative maintenance.

Disadvantage: Cost of spares kit, cost of personnel

Note: To select the spares for a machine the manufacturer gets an engineering opinion on the MTBF by component. This estimate can be used to select a spares kit which will solve the majority of problems. The full spares kit would include every possible component, the reduced spares kit will solve less than 100% of the possible problems but will cost less. The trade-off is how often will the failure occur versus how much does it cost to stock the part to repair the hardware.

If loaners are provided the downtime of the professionals will be minimized. The loaners can be provided while the repairs take place. Who does the repair will also affect the cost of the the maintenance. The manufacturer is usually the lowest cost repair option but also if done through depot service will have a turnaround of up to 1 week. If there is a warranty the manufacturer will require that the repairs are done through authorized repair centres or the warranty will be voided. The offsite versus onsite service will also make a difference. If a service company offers onsite service there will be a built-in cost for the technicians time, wear and tear on the vehicle, cost of stocking spares, cost of tools etc.

The key item when selecting the service option is to ask the question "What does it cost to be without the machine for 1 hour, 2 hours, 1 day, 2 days, 5 days etc.?" Knowing the costs associated with not having the machine available to the professional will assist the budgeting process for the maintenance options. This question should be reviewed for each workstation quarterly as the dependence on a workstation often increases as time goes on.

Now that's fine for individual machines but how about an overall plan?

At present there are several different agencies which will repair and maintain various micro computers on a Canada-wide basis. These organizations will offer service on both a time and materials basis or a contactual basis. Onsite service will cost different amounts depending the distance from the suppliers depot. Most service is provided on a "zone basis" where the travelling time is allowed for by a variance in rates. The contract will vary according to where the repair depot is, thus a nationwide contract with any one company has to take into account the possibility of zone charges in each city. The service companies always charge depending on the response time requested, 4 hour response is a standard. The

charges for the service in under 1 hour, 2 hours, 4 hours, 8 hours are the first stage. Then the next question is what hours will you want your service on call for 8,10,12,18 ands 24 hr coverages all cost different amounts.

If the service company does not currently service the machine (or peripherals) you have selected then it will have the cost of purchasing spare parts and training. That cost will have to be spread over the number of contracts it expects to service from that location over the expected useful life of the spares kit. If the only one being serviced from the depot belongs to your organization you will pay for the spares kit. Long term service contracts are usually better under such circumstances.

To understand the economics of having a micro one must understand the costs and benefits associated.

```
COSTS  -  Purchase - cpu
                   - printer  (dot matrix or daisywheel)
                   - plotter
                   - modem
                   - monitor
                   - disk drives (floppy or hard disk)
                   - cabling

                   - software

                   - computer magazines
                   - training courses
                   - conferences/seminars
                   - user groups

          Supplies - disks
                   - paper
                   - ribbons
                   - printheads/printwheels
                   - plotter pens
                   - spare parts

          Labour   - Hardware technician
                   - Software or application specialists
                   - Trainers
                   - end users
```

Your variable costs should be traded off against each other. The trade-offs are based on support objectives.

1) Allowable MTTR 2) Cost of support (max. budget) 3) Cost of spares

The concept of support is actually quite vague unless we have some kind of measures to determine whether the support is good or bad.

A measure of how well the support is going is a machine availibility statistic. Another might be the number of failures and the average length of failure. A log should be kept for each piece of equipment and peripheral so that each piece can have it's own record kept. Remember you can get a lemon.

The aspects of hardware support (a summary)

1. How fast do you want the problem repaired.
2. What hours of coverage do you want.
3. How many different pieces of equipment do you want covered?
4. Where are you going to locate the equipment?
5. Where do you want the equipment repaired? On-site, local, regional, central
6. How many spares kits are you going to purchase?
7. How many loaners will you need and where will you locate them?
8. What is the procedure if a hardware fault is suspected?
9. How is support handled? local,regional,national contracts;employees;warranty
10. How long is a piece of equipment to be supported for? (eg. 3yrs)

11 .Are the non-warranty period economics different from the warranty period.

These factors will change in both importance and cost over time and therefore should be reviewed from time to time.

The computer industry has set up networks to supply and service machines. The manufacturers network, the third party service network and then your own network are the three networks to look at for service for your machine. Just as for the cost of a telephone system you must decide which network is the cheapest to go for. It may well be that while under warranty the manufacturers network provides the least cost service for the time frame desired. Later it may prove less expensive to use third party service.

The use of the existing service networks should be examined before attempting to build your own. If you build your own you will also have to maintain it, staff it buy spare part etc. In a few localized cases where the number of machines in a particular locale is large enough to support it, it may be cost effective to have your own service but in the majority of locations other alternative would be preferable.

Some of the corporations now supplying service for hardware in Canada are:

```
Xerox     - 12 locations in Canada (carry in service)
Aabex     - 17 locations (Electrohome on site       )
Miscoe    - several major cities
Eatons    - 44 locations (2hr service on site     )
TRW       - many locations
Dataforce - Bell locations almost everywhere.
CGE        - also getting into micro software sales and printers
```

To determine the costs of using any one of these organizations one must first know how many of what brand of machine will be located where? To attempt to determine the strategy at this point in time would not be effective. The best that can be done is to develop a pro-forma breakdown of the expected machine population by location and brand and then obtain pricing information from each one.

The role of the Micro-centre

The micro centre has several functions:

1) To assist in acquisition of micro computers
2) To assist in acquistion of software, hardware and peripherals
3) To act as an advisor to end users wishing to acquire micros
4) To support the end users and assist in solving any problems which might arise regarding the micros.
5) To maintain lists of tested software as well as reviews both by the centre's staff, end users and trade magazines.
6) Maintain a library of relevant DP publications.
7) Co-ordinate both hardware ans software support.

A step-by-step approach to the acquisition process would be as follows:

1. Statement of requirements to be prepared.
   This can be filled in by the user using a checklist provided by the microcentre; Microcentre staff will be available to help at any time should the user desire.
   Statement of requirements to include expected benefits, data volume estimates, communication requirements, implementation schedules, anticipated growth.

2. Examine Feasibility of proposal in light of
   a) Alternative solutions (minis,mainframe,existing manual systems)
   b) Estimated cost versus benefit
   c) Organizational policies
   d) Security requirements
   e) Manpower required to install, support and maintain.
   f) Importance of project to organization as a risk factor should the project fail or data be lost or destroyed etc.

3. Make a preliminary recommendation to the end user. Include a copy of the feasibility findings and all caveats.

4. Assist the user in a software selection. Be sure to select for all projects to be run by this enduser as the software ought to run on the same hardware.

5. Assist user with hardware configuration plan including:
   a) RAM requirements - amount of main memory
   b) Output requirements - printers, plotters, monitors
   c) Data storage requirements - hard disk, floppies
   d) Communications equipment requirements - modems, LAN links
   e) Workstation furniture - desks, acoustic covers etc.

6. Assist user with software configuration.

7. Assist user with implementation schedule, include installation, training, allow for lead times as not all items are instantly available.

8. Assist user with vendor selection; include hardware maintenance recommendation if applicable.

9. Purchase equipment.

10. Follow implementation schedule to get user up and running.

The micro centre will provide configuration support by testing any proposed configuration. The software can be supported by having the micro centre purchase tutorial learning packages, and sponsor courses. Once users become familiar with the packages, the organization should develop an informal user group for each package so the the users of the package can swap ideas. To facilitate this an electronic mailbox system could be set up so the users could browse hints, suggestions, known problems and work-arounds. This way a user could try to solve the problem via self-help. The first line of defense for a user would be the micro users in his office followed by the micrx@ centre. A list of subject matter specialists should be maintained, and these persons encoraged to contribute their ideas, suggestions and even to teach on behalf of the micro-centre.

Any problems referred to the micro-centre would be documented and the fix put on the bulletin board. Part of a new users "getting started" kit would included a hardcopy of the bulletin board. If the microcentre has trouble solving a problem, the micro centre will go to the manufacturer or software supplier for help. This way the micro centre can maintain the known problems, helpful hints etc. on the bulletin board.

The micro-centre should have in its budget the funds to purchase magazines, join user's groups and attend certain conferences.

Attendance at a conference would be based on certain criteria being met:

1. That the conference is relevant to the hardware or software environment.

2. The attendee is given five of the problems which are currently under study by the micro-centre and is briefed on the background. Part of the attendees job is to come back with the solution and the phone numbers and addresses of the persons who provided the relevant information regarding the problem.

3. The attendee writes a report on the portions of the conference which were relevant to his/her position. The attendee should conduct a presentation within 1 week of the conference to the members of his operating group.

4. There should be a rating form to be filled out evaluating the conference/seminar as a whole, and noting any particularly good speakers , ideas or features.

This way the micro-centre budget could be used both by the micro-centre and any users of the micro-centre.

The micro-centre will provide support to a limited number of hardware and software alternatives. Those alternatives which are not on the supported list will be acquired only against the recommendations of the microcentre. The justification for purchasing a non-standard

alternative must be submitted as part of the reqirements definition. It is probable that certain engineering packages are only available on certain brands of hardware and therefore if after

confirming investigation no supported alternative is available then the purchase

would go ahead. The micro centre will determine with the user what level of hardware support is necessary for the system and will if requested arrange for the necessary contracts to be negotiated. The micro centre has a responsibility to assist the justified non-standard equipment user.

The organization must consider what to do about existing equipment which does not qualify as supported equipment yet serves a useful function. How long will these systems receive support? This decision will be partly based upon the support required by these systems.

Software Support

Training (classroom)- it has been estimated that it takes 100 to 200 hours to master the personal computer for a managers needs. The use of classroom training allows some of the frustration for managers and executives to be cushioned. The trouble is that not every application can get the instant results which are so evident with the simple spreadsheets which are usually the first application the personal computer is used for by managers or executives.

Tutor programs - this allows self-paced learning which is sometimes a less embarassing way for executives to get familiar with PC's and their uses.

Binders of application notes by software package shipped for installation. - these assist the new users to avoid the same problems other users had. By including templates for the most frequent application of the software package, the learning curve can be shortened.

Organizations hotline - if a user does get stuck there can be someone there on the other end of the phone to help out.

Hints & tips published from time to time - as more and more is placed on personal computers it will become easier for the micro-centre to identify the common problems and let the organization's personal computer users know about how to avoid them.

Manufacturers hotline - to be used by the micro-centre. All major problems should have the micro-centre as a middleman in order that they learn the problems and can collect the answers. In the case of extreme difficulty the micro-centre should set up a conference call with the user and the manufacturer (and listen in), to help solve the problem.

Grassroots - the best form of support will be through the users themselves. Application specialists should be tapped to assist the micro-centre whereever possible.

Conclusion: It must be remembered that the micro-centre plays a vital role in supporting professionals. The support is in several forms but the entire purpose is to allow the professional to get on with his/her job and not have the productivity tools slow him down.

Birket Foster is based out of Ottawa. He was instrumental in developing and implementing the initial marketing strategy for Quasar's QUIZ product. He is chairman of SIG-SOFTVEND the special interest group for software vendors. His company provides marketing planning and support to software vendors and software selection services to users. THE CATALOGUE, a comprehensive publication describing the third party HP3000 software is published by M.B. Foster Associates Limited. This paper was word processed using a HYPERION Personal Computer which uploaded it to an HP3000 using its built in software and modem. The HYPERION was designed and built in Ottawa to be IBM-PC compatible by the BYTEC group.