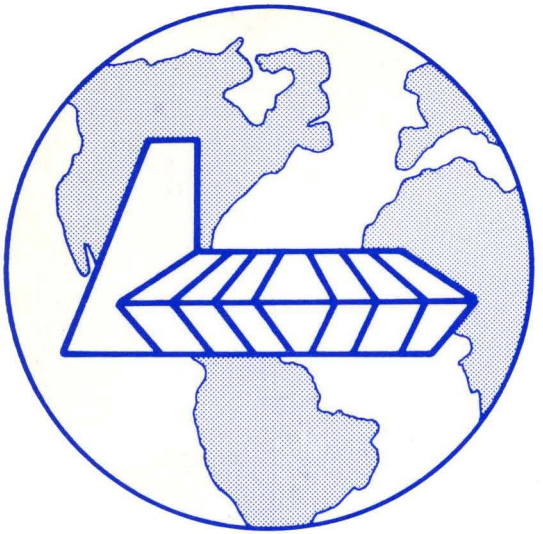


HP 3000 INTERNATIONAL USERS GROUP

1983 HP 3000 INTERNATIONAL CONFERENCE
MONTREAL APRIL 24–29
QUEEN ELIZABETH HOTEL
MONTREAL, QUEBEC, CANADA

PROCEEDINGS

CONFERENCE COMMITTEE

Denys P. Beauchemin . . . . . .Conference Chairman

Myer Kwavnick . . . . . .Conference Vice-Chairman

James B. Fairchild . . . . . Conference Manager

Hank Van Leeuwen . . . . . Paper Selection

STAFF

William Crow'. . . . . Executive Publisher/Association Manager

Renaye Lee . . . . . Conference Manager

John Knapp . . . . . Publisher

Sandra Hawker . . . . . Managing Editor

INDEX OF PRESENTATIONS — by SPEAKER

INDEX OF PRESENTATIONS — BY CATEGORY

COMMUNICATIONS

DATA BASE SUPPORT

LANGUAGE SUPPORT

MANAGEMENT

PERIPHERAL SOFTWARE

PRODUCTIVITY TOOLS

SYSTEM MANAGEMENT

USER APPLICATIONS

HP3000 DATA RECOVERY

David R. Beasley
Systems Engineer
Hewlett-Packard Company

Everyone who manages a computer installation realizes that no matter how well you schedule system backups, there will inevitably come a time in which the backup tapes have parity errors, or the disc has a head crash or drive fault, or the system crashes prior to the daily backup, or someone simply halts the machine on the way up from a COOLSTART! Each of these cases present some unique problems to overcome, but one common problem is that valuable data may be lost. Or is it? Has the data been physically destroyed, or has MPE become logically inoperable due to the corruption of some key data structure such as the system directory? Most data can be "physically" recovered given enough time, the know how, the proper tools for the situation, the equipment, and the patience. There are certain questions to be asked in all situations involving "lost" data. Can the data be re-entered from scratch? How long will it take? Can the data be physically removed from the storage media? Can you afford to lose the Data? Is the amount of time required to recover the data more valuable than the data itself? Once you have decided that an attempt must be made to retrieve the data, you must understand how the data is organized and what tools are available to help you in saving all or part of the data.

There are many different ways to organize data, such as IMAGE files, KSAM files, or simply a standard MPE file. In many cases, this is a critical factor in deciding how much effort should be spent in an attempt to get the data back. Because of the sometimes complex interrelationships between the data, it is not always "good enough" to retrieve some, but not all of the data. For example, saving a KSAM key file, but failing to save the corresponding data file may not gain anything. Another example is when you fail to restore one of IMAGE's dataset files. At first, you might think that all you need to do is get that one file from a previous backup tape set, but that could leave you with a logically corrupt data base. In such situations, it may be necessary to accept the loss of one file, or to even "bite the bullet", RELOAD MPE, and re-enter the data. It is for this reason that transaction logging and saving daily transactions in hard copy format are sometimes recommended. Let me add a word of caution to those users who do partial Sysdumps on a daily basis. If you have IMAGE data bases, all of the files may not be STORE'd to the tape because some of the datasets may not have been accessed. This is ok unless you run into trouble later on down the line trying to re-construct your data base due to lost data. If you are extremely careful,

and if you make sure that you restore the files in the correct order, you will not have a problem, but why give yourself a chance to make an unnecssary mistake. It is recommended you do a DBSTORE of your databases to keep all of the datasets together. Some people may argue that this takes more time. This is true, but how much time do you have to recover you database if a failure occurs? Each application is different, so thinking of these issues, and planning for disaster recovery will help you avoid losing critical data.

Although data can be logically organized in a variety of ways, it is simply a "bunch of bits" to the computer hardware. This fact is not new to anyone who understand computers, but is it vitally important to remember when you need to recover "lost" data! The mass storage devices such as the disc drives and tape drives do nothing more than physically record the data that the software tells it to. These peripherals aren't responsible for understanding the logical organization of the data. Suppose that MPE can't find a particular file because the area of disc occupied by the system directory is unreadable due to a bad track. The file is still on the disc, it is simply unaccessible in the logical way MPE expects to locate it. There are utilities for such a case to assist you in retrieving that file. (You may need to write your own sometime). It is almost always possible to retrieve the physical record of data from the peripheral and its media, unless of course the media has been mutilated and destroyed. However, if the peripheral was broken at the time in which the data was written to it, the data may be garbage when it is read back. In this situation, you need to evaluate whether or not the data can be "repaired" once physically retrieved or is re-entering the data the best approach.

Assuming that you have made the decision to attempt to retrieve the data from the physical media as opposed to returning to a previous known good copy of the data and re-entering your work, there are several useful utilities available to you. Some of these are supported by Hewlett-Packard and others are not. The syntax of the utilities will not be focused on since these are documented in the Hewlett-Packard System Utilities Manual (30000-90044) or in other places such as the User Contributed Library. For most of these utilities, an understanding of the HP3000, equivalent to that which is taught in the System Manager's class offered by Hewlett-Packard is sufficient to allow the user to use these utilities with confidence. During the discussion of the utilities and the situations in which they can be used, an understanding of the disc organization and some key disc resident data structures must be understood. You may refer to the MPE System Tables Manual (32002-90003) for a description of these table layouts. (Appendix A provides table layouts for some of these disc resident tables).

What can be saved? Disc files and files from bad SYSDUMP/ STORE tapes are the most common types of files necessary to recover. Let's discuss disc files first. Disc files are located by MPE by finding a pointer to the LDEV and sector address of the file label. (Note that the LDEV pointer is really a volume table index). Each file has a file label which contains a description of the characteristics of the file such as the record size, the block size, the EOF pointer, etc., and it also contains an extent map which points to the LDEV and sector address of the other extents which make up the entire file. Note that all extents do not necessarily reside on one particular disc. If the system crashes and catches you without a good set of backup tapes and if MPE is logically inoperable due to data structure corruption, a utility called SADUTIL can be very useful. SADUTIL stands for Stand Alone Disc Utility. It is supported by HP, and is documented in the System Utilities Manual. SADUTIL does not run under the control of MPE. It is cold loaded similarly to the way MPE is loaded as a stand alone utility. SADUTIL will allow you to save files to tape which otherwise would be lost if a RELOAD of MPE was necessary. SADUTIL expects some data structures on disc to be intact, however. Each mounted system volume must have a good volume label, (sector 0). If the system directory is intact and the correct address of the directory is valid in the Cold Load Information Table, (sector 28), and if the Volume Table is good, you will be able to save files with the @.@.@, @.@.acct, or @.group.acct option. If the above data structures are invalid, you may attempt to re-build them with the EDIT functions in SADUTIL. Another option would be to use the FIND command which scans the entire disc looking for file labels. There is no guarantee that the data will be valid, however. After successfully saving the files to tape, a RELOAD of MPE will be necessary along with any other appropriate action required to correct the original problem. Once MPE is running again, a program called RECOVER2, which is also supported by HP, can be run to retrieve the files from the SADUTIL tapes and re-create them in the system directory. Note that the data in the files is not guaranteed to be valid and garbage free!

While we are on the subject of SADUTIL, let's consider a case where someone halts the system during a system start up, such as a COOLSTART or COLDSTART. INITIAL is the program which, when bootstrapped into memory, executes to build MPE and MPE's tables and data structures. There are several disc resident data structures which INITIAL depends on (unless doing a RELOAD) such as the Cold Load Information Table. INITIAL locates other critical data structures on disc from this table. Some examples of the disc resident tables required are the system directory, the Volume Table, and a file named CONFDATA which contains configuration information. The format of these tables are in the System Tables manual.

In order to protect the integrity of the operating system, there is also something known as a Cold Load ID. This Cold Load ID is incremented by INITIAL on each start up in order to ensure that all volumes belonging to the system are mounted together as a set. This Cold Load ID is kept in each system volume label, (sector 0), in the Cold Load Information Table, (sector %34), and in the Volume Table. This Cold Load ID is also kept in each file label. When FOPEN opens a file, the Cold Load ID in the file label is compared with the "current" Cold Load ID so that the file system will know if certain kinds of information are current, such as; is the file currently open and shared?; is the FCB vector valid?; are the STORE bits valid?; etc. In this way, FOPEN knows if the file was open during a system crash. If you halt INITIAL during a start up, the Cold Load ID's may get out of synch and INITIAL will not allow you to subsequenty start the system, but instead will give you the infamous message of "MOUNT CORRECT VOLUMES OR RELOAD". With SADUTIL you can EDIT the discs in the correct locations to get the Cold Load ID's back in synch. The precise locations are in word 7 of each system volume label (sector 0), word %12 of the Cold Load Information Table (sector %34), and in words 1 and 3 of the Volume Table which is pointed to by words %124, %125 of sector %34. By modifying these words, you may save valuable data by allowing MPE to be restarted and a STORE to take place. It should be noted that anytime you have to "rebuild" any table for MPE, it should only be done with the intent to save the data. Since you can never be certain as to what else is not correct, a RELOAD should always follow this procedure to ensure operating system integrity.

DISKED2 is another supported utility which can be used to read or modify any area of the disc under the control of MPE. However, DISKED2 is not under the control of the file system so you must use this utility with great caution. FLUTIL3 is an unsupported utility which will allow you to modify certain words of a file label if it becomes corrupt for any reason.

Let's turn our attention to bad SYSDUMP/STORE tapes. It is not uncommon for tapes to have several parity errors or to develop "bad spots" over time. GETFILE2 is a very useful, although unsupported utility to help you read past bad spots on a tape and recover files that MPE's RESTORE will not allow you to get. Another scenario for which GETFILE2 is very handy is as follows. Suppose someone does an FCOPY of a small file onto one of your STORE tapes by mistake. Well, obviously, the data which was overwritten is lost, but the data beyond the new logical end of tape is still there even though RESTORE does not recognize the format of the tape. GETFILE2 will allow you to read the entire tape looking for files labels. FCOPY is also a good tool at times to recover files from tape.

There are a couple of other unsupported utilities that I would also like to mention. STAN (Store tape analyzer) and TAPLIST. Either of these can be used to look at the directory at the beginning of a STORE tape if you're not sure what's on the tape, or if you're having trouble RESTORE'ing a file because of an incorrect accounting structure.

One more utility that you should be aware of is IOCDPN0. This program is not for the novice and will require a thorough understanding of MPE's I/O system. IOCDPN0 allows that user to specify any or all of the parameters to ATTACHIO. ATTACHIO is a procedure in MPE which interfaces the file system to the I/O system. With IOCDPN0, you can totally bypass the file system, and you can request any function code that the driver recognizes for whichever device you are accessing. If the data you are trying to recover can be physically read from the media, IOCDPN0 will let you do it. Let me emphasize that great caution should be used with this utility. A "typo" could be catastrophic even when you are simply reading from a device. For example, if someone is FCOPY'ing a tape file, and I accidentially specify the wrong ldev, (the tape drive), in my parameters to ATTACHIO, I will have just "blown away" the user doing the FCOPY. He'll end up one record short. It could be worse than this. Suppose I'm trying to read the volume label, (sector 0), of the system disc, just because I'm the curious type. If a write function is specified by mistake, it could be RELOAD time tonight! This utility is very powerful, but very dangerous!

It is beyond the scope of the paper to present all of the information you need to throughly understand in order to be considered a "data recovery specialist"; however, I hope you have been somewhat enlightened as to the types of issues that must be considered and what types of tools are presently available to assist you. Remember that planning for data recovery and a good "prevent defense" will save more data than all the tools you can imagine!

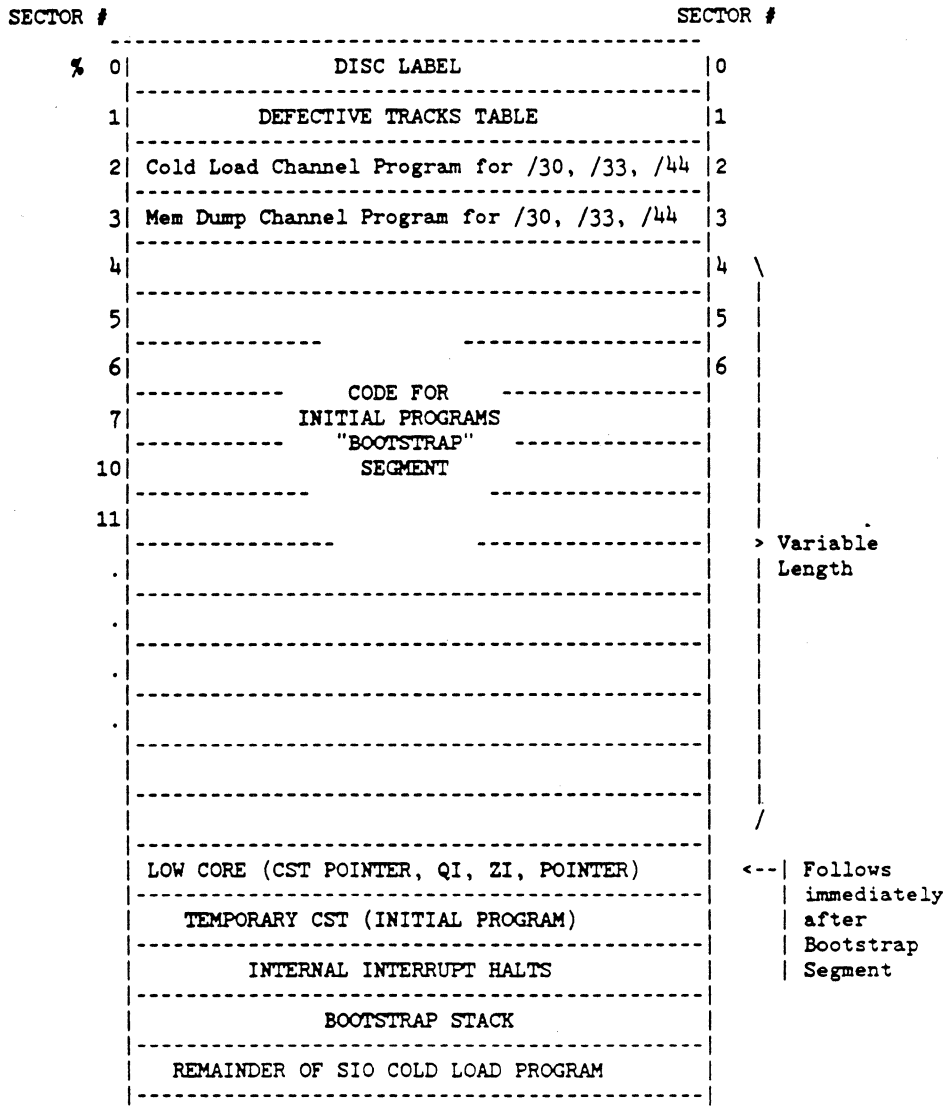
Appendix A

(Excepts from System Tables Manual)

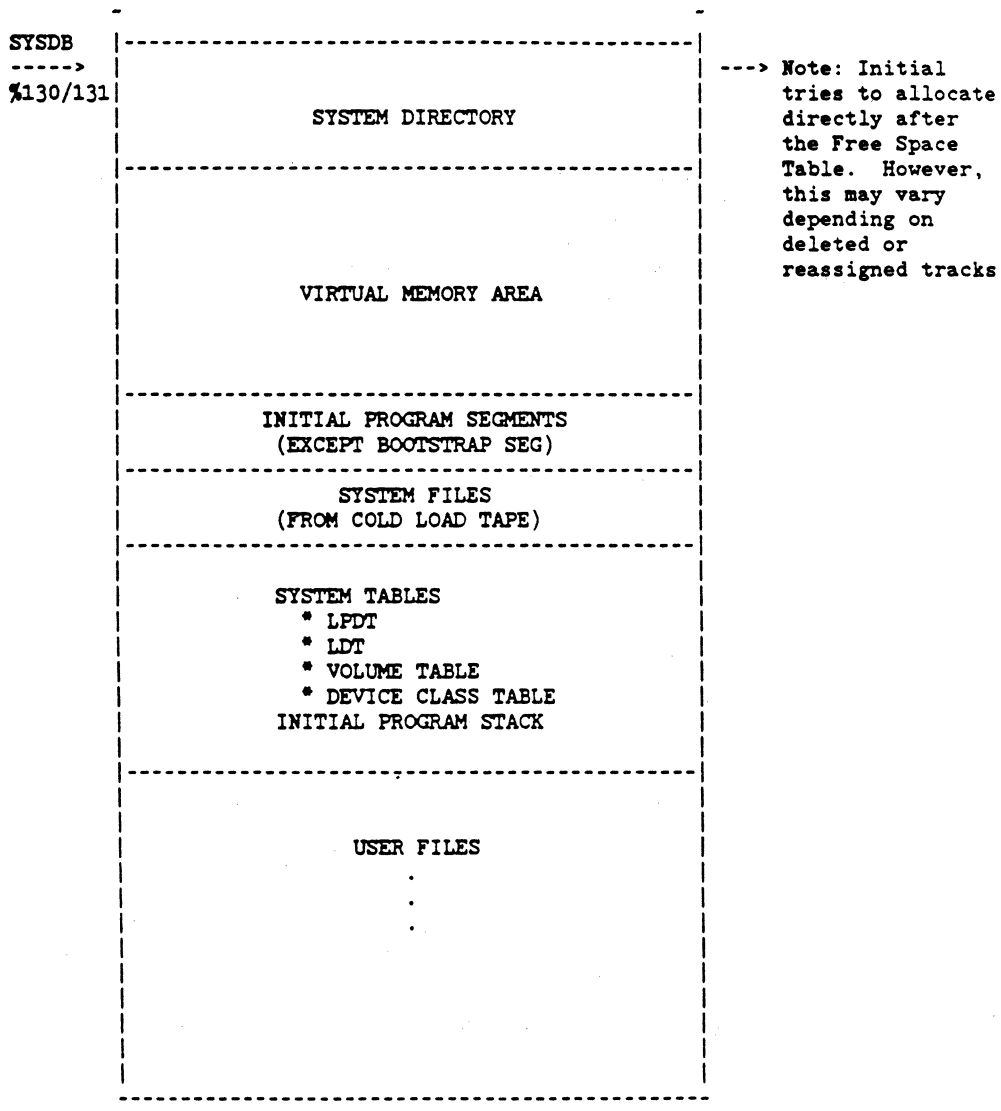
(P/N 32002-90003)

CHAPTER 3 DISC LAYOUT

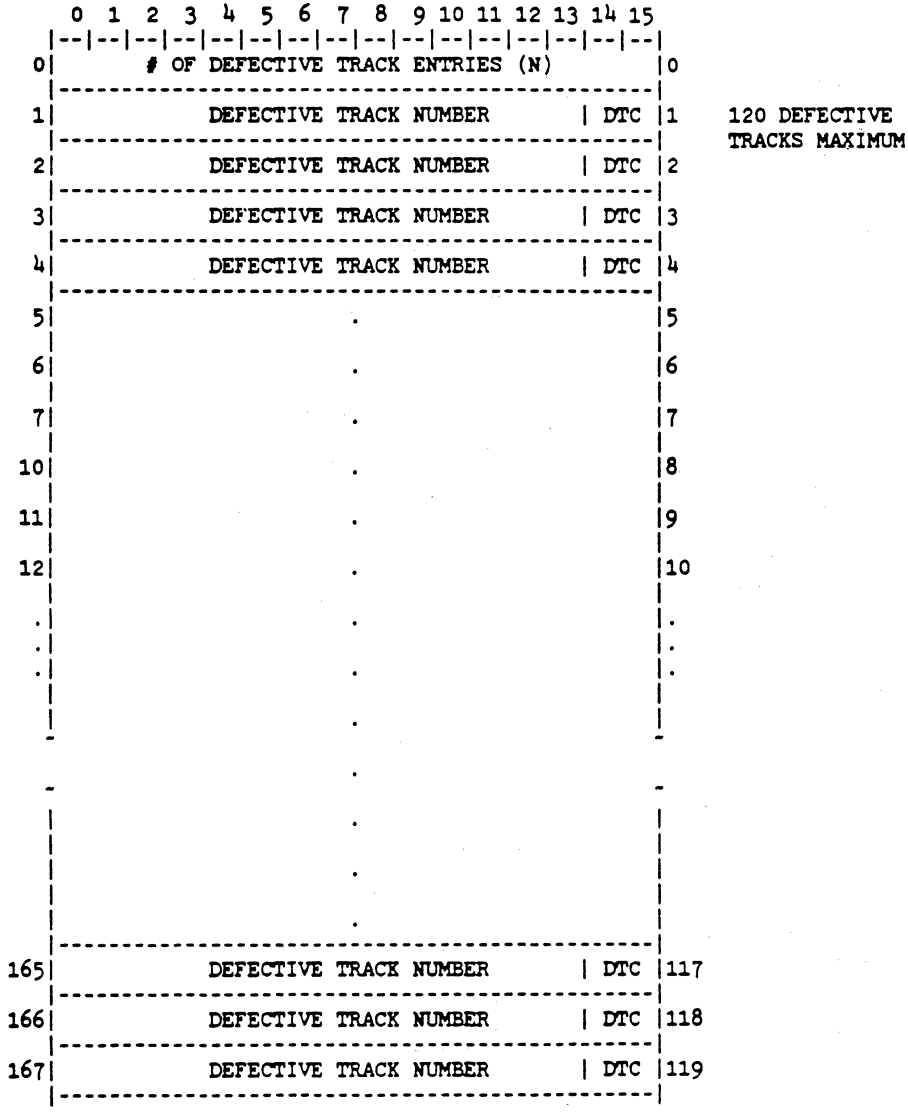
SYSTEM DISC LAYOUT



SYSTEM DISC LAYOUT (CONT.)



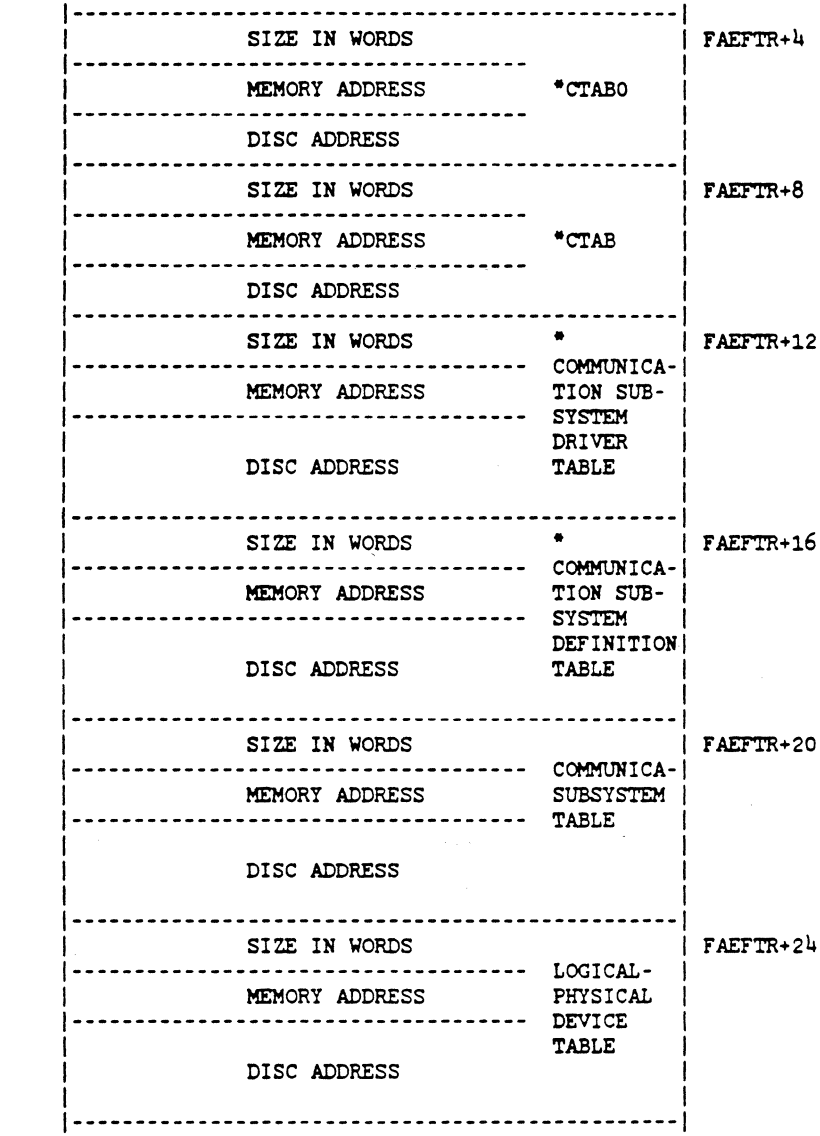
DEFECTIVE TRACKS TABLE (Sector 1 of Disc)



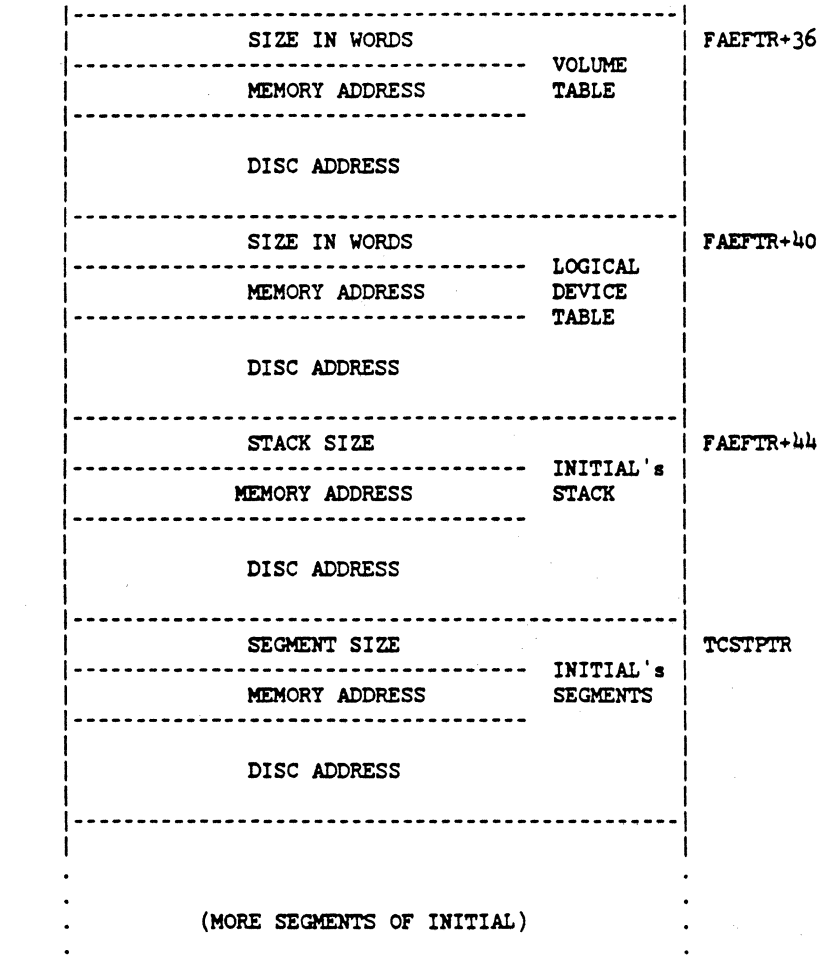
DISC COLD LOAD INFORMATION TABLE (SECTORS 28-29)

0	pointer to table information	FAEFTR >----------
1	pointer to temporary CST info	TCSTPTR
2	# of entries to read on disc cold load	NREAD
3	# of code segments in INITIAL	NVTCST'
4	INITIAL's DB value	INITDB
5	INITIAL's DL value	INITDL
6	INITIAL's Z value	INITZ
7	INITIAL's Q value	INITQ
8	INITIAL's S value	INITS
9	| SYSDISC type | subtype	DISCTST
10	cold load ID	COLD'LOAD'ID'
11	log file number	LOG'FILE'NUM'
12	directory disc	
		DIRADR
13	address	
14	ldev 1 virtual memory	
		VIRMEMADDR
15	disc address	
16	# LOG PROCS	
17	LOG ID's	
18	RIN table	
		RINADR
19	disc address	
20	directory size	DIRSECT
21	#sectors in virtual memory region of LDEV 1	SECTORS IN LDEV1 VM
22	UNUSED	
23	RIN table size	RINSECT
24	# of RINS	RINS

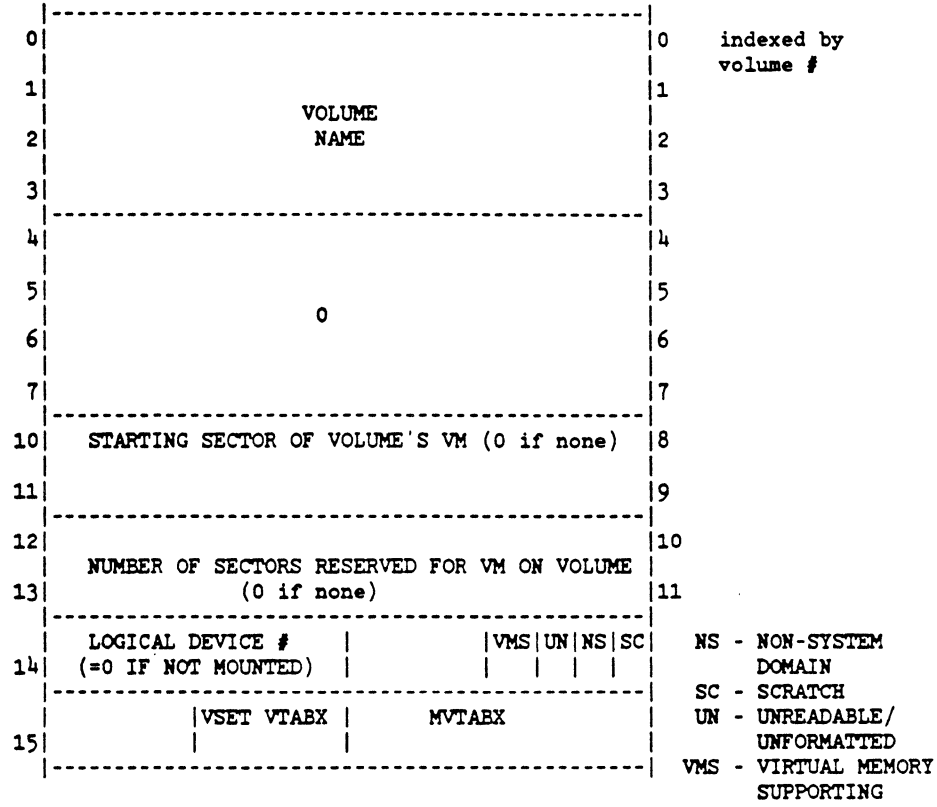
DISC COLD LOAD INFORMATION TABLE (CONT.)



DISC COLD LOAD INFORMATION TABLE (CONT.)



TYPICAL SYSTEM VOLUME ENTRY



Another IMAGE Article

by Joseph Wm. Berry
Consultant
1800 S. Robertson Blvd.
Bldg 6, Suite 2000
Los Angeles, CA 90035

Introductory Remarks

Why indeed do we need another article about IMAGE data bases? In the last few years, many articles have been presented both in the local, national, as well as international HP Users' Group Meetings. By this time everyone knows that capacities should be prime, that alphanumeric keys are generally better than numeric keys, etc., etc. With respect to that statement, I would like to make two remarks. First, having been a systems specialist for Hewlett-Packard trained in both performance consulting as well as data base consulting, time and time again I have seen users making these same mistakes when implementing their data bases. I am referring even to "knowledgeable" users. It is my opinion that a majority of all performance problems can be traced to bad data base design, bad data base design implementation, or badly designed/implemented programs that access a data base. Second, there have been very few tools available to the user community (or for that matter to Hewlett-Packard systems engineers) for analysis of data bases in order to verify suspicions that a data base was a significant factor in performance degradation.

As a result of this feeling of hopeless frustration, knowing in one's heart that the data base was at fault, but being unable to prove it, three systems engineers wrote a program known as DBSTAT2. This program is not to be confused with other programs with similar names (e.g., DBSTAT in the users' library that is written in FORTRAN). Very simply, this program presents a detailed pictorial of the contents of a master data set as well as of a detail data set, along with various statistics about the contents of the data set. This program was authored by Ed Splinter (of InfoCraft Inc.), Ted Dickens (HP, North Hollywood), and Joe Berry.

In the next few pages, I would like to describe some of the uses I have made of this program to help identify user data base problems. It will be obvious (if it isn't already) that this program serves a real need. Unfortunately, it does require the use of privileged mode (PM) capability. And as a result, Hewlett-Packard does not warrant the use of this program on anyone's system (too bad!).

Performance Problems?

There are three major resources on the HP3000 that are potential limiting factors to good performance. These resources are (1) CPU speed, (2) memory, and (3) I/O throughput. With the release of the HP3000 Series 64, Hewlett-Packard has introduced a CPU-powerful computer to the user community. For companies requiring CPU intensive operations, such as solving differential equations, etc., the horsepower of the series 64 is available to provide such solutions.

What about memory? The Series 40/44 can be configured with up to four megabytes of memory while the Series 64 comes with as much as eight megabytes of memory! That's quite an improvement over the days when the maximum amount of memory available was two megabytes. Again hardware presents an easy solution to what used to be a rather difficult problem. Of course, if you don't wish to spend the money on the extra hardware, then some work, usually programming, will be required. But, at any rate, there is a fast "dollars and cents" solution.

When it comes to disc I/Os, the picture changes somewhat. Hewlett-Packard has not announced any markedly improved disc I/O devices for its products. This is not to say that one cannot get a fast number of I/Os through a system. I have personally seen HP3000 systems producing 60, 70, and over 80 I/Os a second on live applications. Unfortunately, this is not the norm. Whenever I see an HP3000 that is I/O bound (again, which is most of the time), I immediately begin by examining the data bases that are in use (in one notable instance, however, the user presented me with an I/O intensive KSAM application!).

History Revisited

In order to properly appreciate DBSTAT2, I will present some of the particulars of its history. In the spring of 1980, an HP OEM company in the downtown Los Angeles area requested assistance from HP. They explained that one of their user sites was having particularly bad performance problems. They requested that HP send someone to their site in order to help identify this HP "IMAGE bug". Their

specific problem was this: certain transactions, which performed DBPUTs to a manual master data set, were taking an extraordinarily long time to process.

It did not require a very sophisticated program to identify that this data set was 99% full! (This is a fairly "knowledgeable" customer with five HP3000s. Just how production data base data sets "happen" to be at 99% full is beyond me. Doesn't anyone monitor these things?) Our suggestion was rather obvious: increase your capacity or decrease the number of records. The customer explained, however, that they had done this before and that decreasing the number of records not only did not improve response time, it actually worsened the response! This indeed was difficult to understand. The "slower" response time was demonstrated with a stand-alone batch job that didn't have subjective response times associated with it.

So what was the problem? We developed a theory we thought would explain the evidence. But how would we verify it? And furthermore, how would we demonstrate to the user that it was a data base design problem?

And so, along came DBSTAT2. The first illustration (fig. 1) at the end of this article shows the user's data base when it was 99% full. Details of the output will be explained later. Note, however, that very few blank spaces exist for new records to be inserted. The second and third illustrations (figs. 2A and 2B) show how the data set appeared when it was 93% full. There are a number of interesting things to note with respect to this output and the previous one. First, it is very clear that the key used in the data set didn't agree very well with IMAGE's hashing algorithm (that is, the records weren't randomly distributed throughout the data set). Also note the large number of secondaries (they're identified as dashes) occupying the first 2,000 records. The user had told us that on a regular basis they delete old, dated records from this data set. What we were seeing were the results of this cleaning-up process. However, by comparing the two illustrations, it is clear that some records had been added to the data set before DBSTAT2 had had a chance to examine it. This accident provided us with the required clue to determine why the user's DBPUTs were now taking longer than before. A final analysis of this example will be presented further in the paper.

Operational Characteristics

Permit me to digress somewhat by explaining some of the operating characteristics of DBSTAT2. Because of the way the program accesses the data base, exclusive use of the data base is required. Many users have found it difficult

to part with their system for the few minutes (or hours) that DBSTAT2 requires. In order to make the program easy to use, I have implemented a batch mode that requires only the name of the data base as an input value. The program makes its own assumptions about what to print out. An example of this is as follows:

```
!JOB MGR.ACCT
!RUN DBSTAT2.UTIL.SYS
DBNAME.GROUP
!EOJ
```

Wouldn't it be nice if all programs executed so easily? In this batch mode, one or more pages will be printed for each data set in the data base. Since it frequently happens that the user (or systems engineer) wants to see the output as it is generated, DBSTAT2 spools each data set separately. Therefore, in order to avoid many header and trailer pages, a HEADOFF 6 command by OPERATOR.SYS is recommended.

Master Data Sets

I will now describe some of the display fields presented by DBSTAT2. Look at the next illustration (fig. 3) while reading this text. (The first DBSTAT2 illustrations were from the original version of the program. A number of changes have since been incorporated.)

Some identification information is presented at the very top of each data set. Following this is the field "% SPACE USED". This is the ratio of "# OF ENTRIES" to "CAPACITY". Simple enough. What should this number be? Most HP IMAGE classes say that this ratio should not exceed 75-80%. HP's Materials Management/3000 package (MM/3000) automatically warns the system administrator when master data sets exceed 60% full! How serious a problem is this? The simple answer is, "It is serious!" Let me explain with another example. I was once asked to examine an application program that was purchased by a customer to determine why response on their Series 44 system was so slow (even with just a handful of users). The customer explained to me that the problem was in a program they had recently acquired. He wanted me to "prove" that this application was poorly designed/written. He explained that the problem was not in the data base. I monitored the application and discovered that for each order that was entered (this was an order entry package) something between 200 and 2,000 (usually around 1,500) physical I/Os took place! No wonder the response time was so slow. I further discovered that all the CPU time in the application was due to one DBPUT (courtesy of APS/3000). At that point, I knew the problem was in the data base itself. I executed DBSTAT2, examined

the "# OF ENTRIES," and discovered the master data set associated with the detail that the DBPUT was writing to was 98.7% full! Had I followed my usual procedure of immediately running DBSTAT2 when a data base is involved, this problem would have surfaced much earlier.

Another very important item is the blocking factor of a data set. Since, in general, master data sets are accessed randomly and detail data sets are accessed serially, it is logical for master data sets to have small blocking factors. Why read a large block of data in memory when all you need is one record? If, on the other hand, this data set will be frequently accessed serially, then a large blocking factor is in order (if you will really perform a frequent number of serial reads of a master data set, especially a large one, then I would be suspicious of the design strategy for the data base itself). One must remember that the MPE file system (including IMAGE) reads a minimum of one sector (256 bytes) at a time. Therefore, if the record size is small so that many records can fit into one sector, it is not meaningful to specify a blocking factor so small that the sector isn't completely filled.

If I have a 200-character record, and I assume random access into the master data set, should the blocking factor therefore be one? Is there a need for having it be anything larger than one? The answer, surprisingly, is yes. It is important to realize that with most applications there will be some small number of secondary entries (synonyms) in master data sets. There is a specific exception which I will discuss further down. Assume some number of secondaries and a blocking factor of one. Whenever one of the secondaries is retrieved or whenever a new record is added that creates a secondary, at least two or maybe more additional I/Os will occur because of the block size being one record. As a general guideline, therefore, a blocking factor of three or four should be used unless it conflicts strongly with the 256-byte block size.

An old adage says "A picture is worth a thousand words." In data base analysis, this is particularly true. Each character in the graph of the data set (fig. 3) represents one location in the data set. Three characters are currently used: a blank, a minus character ("-"), and an "I". Both above and on the left side are scalings for determining where in the data set a particular record is located. The blank means that there is no record present (obvious, right?). The "I" means that a record is present and that it is a primary record. Secondaries may have mapped to this record, but they are elsewhere. The "-" means that a secondary record is occupying this location. Note the columns of colons. This character acts as a block separator. An interesting anomoly can be observed in the next illustrations (figs. 4A and 4B). Only along the

left-hand edges of the block boundaries can any secondaries (the minus signs) be found. This demonstrates the way IMAGE allocates secondary storage. In other words, when a record is added to a master data set, and there is already a record located in its proper position, IMAGE begins looking for a free location for storing that record at the beginning of the data block that it mapped into. Many people erroneously believe that the secondaries are stored along side the primary. DBSTAT2 is an excellent tool for verifying such questions.

Beneath the graphic display of the data set are the synonym statistics. Not only are the number of secondaries versus primaries displayed, but also the number of primaries that have one synonym, the number of primaries that have two synonyms, etc. A large number of secondaries is indicative of a potential data base problem.

The last piece of information available is the path information into the detail data sets. DBSTAT2 displays the maximum, average, and minimum lengths of each path and whether that path is sorted. This, too, can be very important. I once performed an analysis on a data base and found that the average chain length into a detail data set was approximately 12,000 records! This wouldn't necessarily have been bad, but for the fact that it was a sorted chain. By making the path unsorted, the customer's 40-hour batch run was reduced to under 5 hours!

To be totally fair, I must correct the implication of the previous paragraph. Sorted chains have a reputation of being bad (especially long ones). This isn't always the case. When a user does a DBPUT to a detail data set that has a sorted path, IMAGE will start by examining the chain in reverse sort sequence (bottom of the chain) first. If the record to be added falls near the end of the chain, only a small number of I/Os may be needed in order to place this record. This is true even if the chain length is many thousands of records long. However, if the records are added in a sequence opposite the sort order in the data set, then IMAGE may have to traverse much of the chain in order to appropriately place the record. This second case is what one needs to watch out for.

Detail Data Sets

Figure 5 presents sample output from a detail data set. The characters used to graphically display the data set are blanks (which means that there is no entry) and "D"s (which means that there is an entry).

Before I explain the value of the detail data set output, a small discussion of how records are physically maintained

in this data set is in order. When a detail data set is empty, records are added in sequential order. If the data set contains 150 records, then only the first 150 physical records contain data. When records are deleted, a "delete chain" in the data set is initialized to indicate where the deleted record is located. When subsequent new records are added, IMAGE first places records by following the delete chain, and only after that chain is empty will IMAGE again place records at the end of the logical file. IMAGE tries to utilize the holes that are created when records are deleted.

Unfortunately, this technique can sometimes result in performance problems. Imagine for a moment the following example: Assume there is a detail data set with five records per block. Let us also assume that the chain lengths are five records long. Therefore, when a set of records is written to the data set, one full block is utilized. On subsequent reads (also assuming that the entire chain is accessed), only one physical I/O is required (assuming no memory contention). This nice situation will continue until a record is deleted. Assume that five records are now deleted (one from each chain). When a new five-record chain is added, these records will not be placed at the end of the file, but rather in the five locations left vacant by the previously deleted records. Therefore, when accessing this particular chain of records, five physical I/Os will take place (one record from each block)! On the one hand, people recommend large blocking factors for detail data sets when accessing entire chains; on the other hand, these blocks cannot be utilized if the delete chain jumps from block to block.

The field "CURRENT EOD" indicates the high water mark for the data set, or how far into the file the data has actually been stored. The field "# OF ENTRIES" means the current number of records in the data set. If these two fields are equal, then the delete chain has no entries. If the delete chain contains record pointers, DBSTAT2 will follow this chain counting the number of blocks read and report its results. It is, of course, quite possible that the same block will be read several times while following the delete chain. If the number of records in the delete chain is large, and if the AVERAGE NUMBER OF FREE RECORDS PER BLOCK READ is small (near one), then the effective blocking factor will also be small. Looking at figure 5 by itself shows that most of the data set is empty. In fact, there are an average of 9.3 free records per block. However, by following the delete chain, we only have an effective blocking factor of 2.1. This does not allow us to take advantage of the data set blocking factor of 10.

To Make a Long Story Short

It was clear from figures 2A and 2B (the master data set that was 93% full) that the key was somehow related to the hashing algorithm. Specifically, the key used was an integer "order number," assigned consecutively. When deletions were made, the deletions represented a range of dates. This explains the sharp band between records and no records. When the user next wanted to add orders to the data set, the specific numbers he wanted to add fell (according to the hashing algorithm) somewhere near the middle of the data set. No empty records were found there. IMAGE, therefore, started examining every block in a forward serial manner. When it reached the end of the data set, it wrapped around and began from the beginning. Finally, the record found a home for itself. As each record was added, all of the holes at the beginning of the data set started to fill up. This was the state of the data set when DBSTAT2 examined it for the second time.

The Grand Finale

I have demonstrated how DBSTAT2 can help identify structural performance problems in IMAGE data bases. An additional feature that should not be neglected is prototyping. When designing a new data base, an important question that should be answered is this: How well will my keys work with IMAGE's hashing algorithm? To find out that the answer is "not too well" after the data set is loaded with one million records is, I think, too late. A safer approach is to build a data set with the appropriate type key and a capacity of approximately 1,000 records. Fill the data set to 80% of capacity and examine the results with DBSTAT2. For the small investment of time expended up front, much time may later be saved.

Acknowledgements

I wish to thank Mr. Dave Stover of the Telephone Employees Credit Union for allowing me time on his HP3000 to continue the development of DBSTAT2. He and Mr. Ted Dickens have given me many suggestions for new features to be included in the software. Lastly, I wish to thank my wife, Galia Berry, for encouraging me to write this article and for helping me in its preparation.

IMAGE MASTER STATISTICS
TUE, MAR 25, 1980, 8:21 AM

DATA BASE: AMS1 DATA SET (MANUAL): ORDERS
CAPACITY: 40001 # OF ENTRIES: 39522
ENTRY LENGTH: 124 BLOCKING FACTOR: 7
% SPACE USED: 99 # OF PATHS: 1

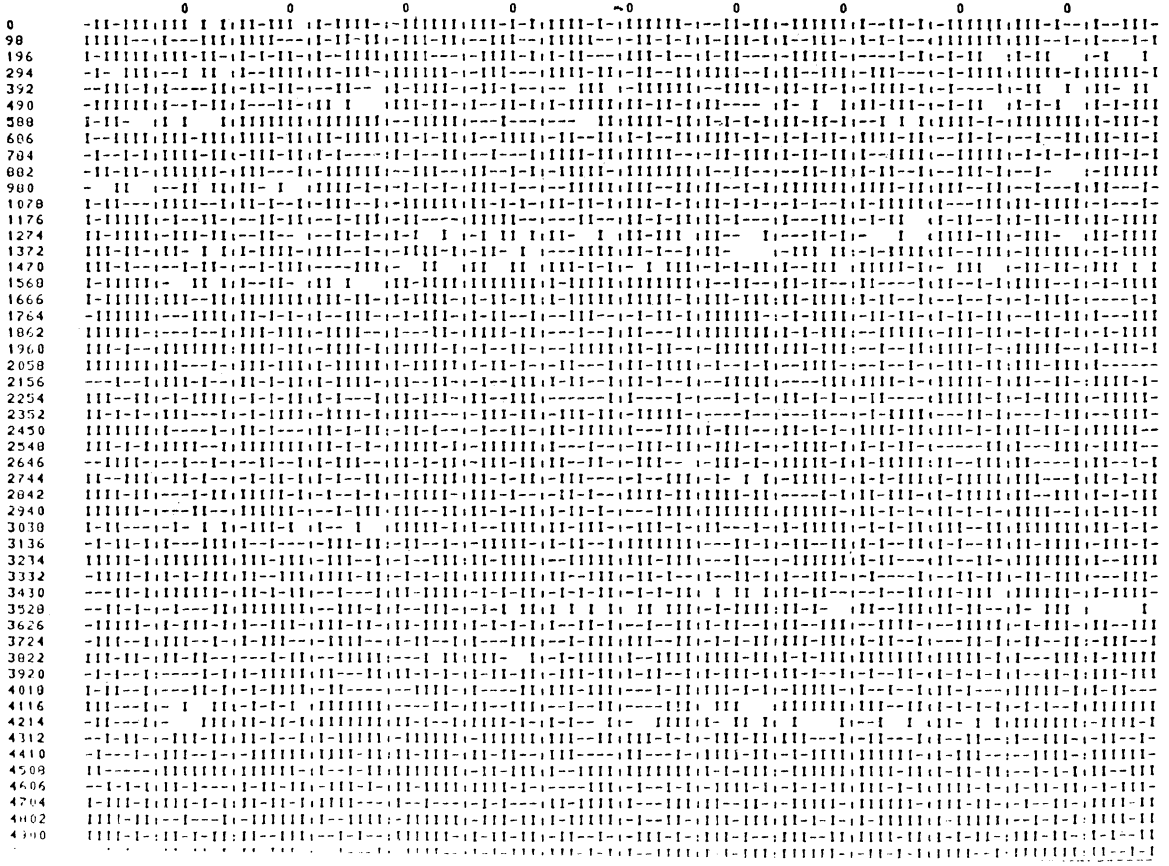
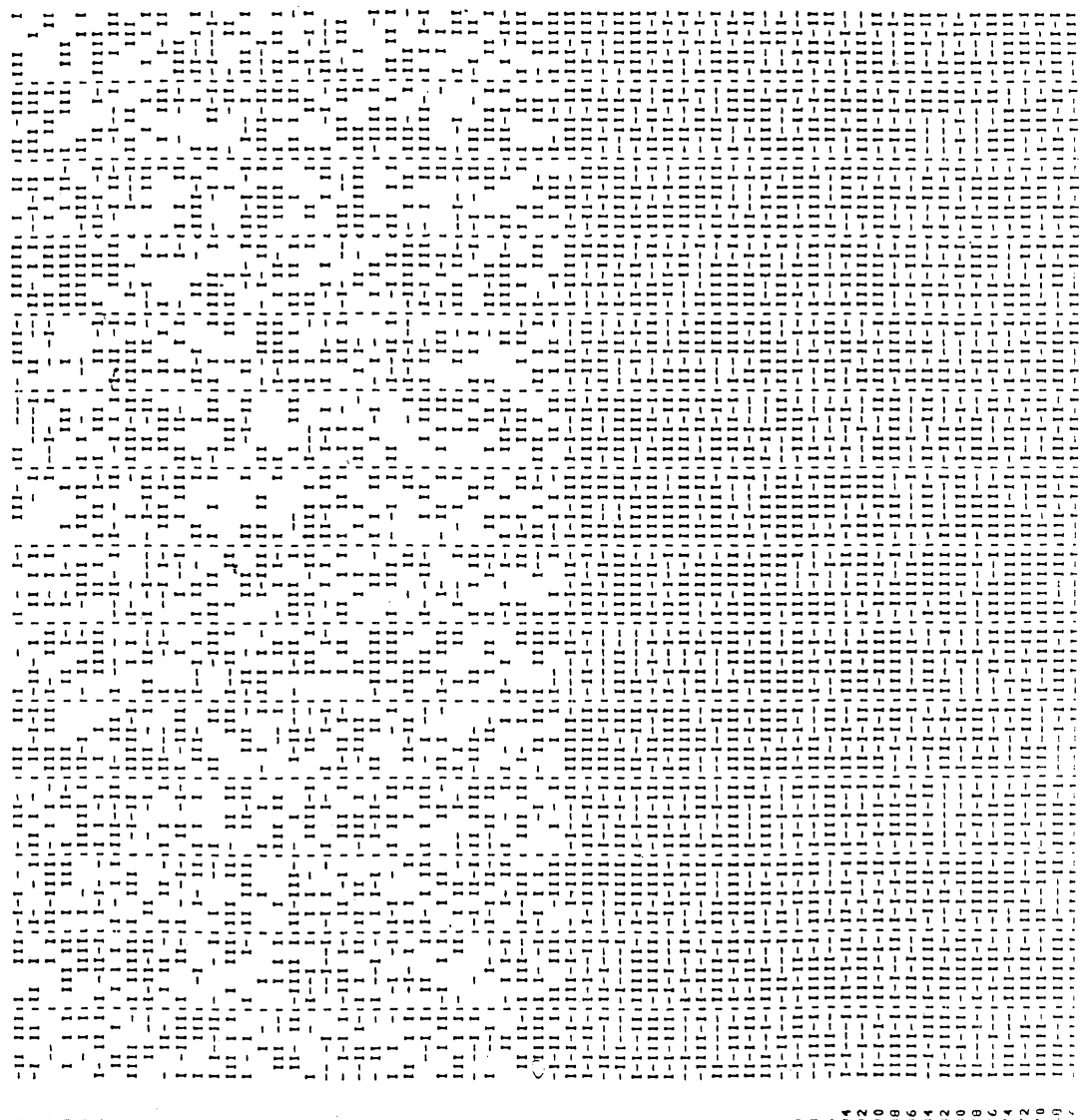


figure 1

IMAGE MASTER STATISTICS
TUE, MAR 25, 1980, 10:15 AM
DATA BASE: AM91
CAPACITY: 40001
ENTRY LENGTH: 124
% SPACE USED: 93
DATA SET <MANUAL >: ORDERS
OF ENTRIES: 37270
BLOCKING FACTOR: 7
OF PATHS: 1
1 2 3 4 5 6 7 8 9
0 0 0 0 0 0 0 0 0
0
98
196
294
392
490
588
686
784
882
980
1078
1176
1274
1372
1470
1568
1666
1764
1862
1960
2058
2156
2254
2352
2450
2548
2646
2744
2842
2940
3038
3136
3234
3332
3430
3528
3626
3724
3822
3920
4018
4116
4214
4312
4410
4508
4606
4704
4802

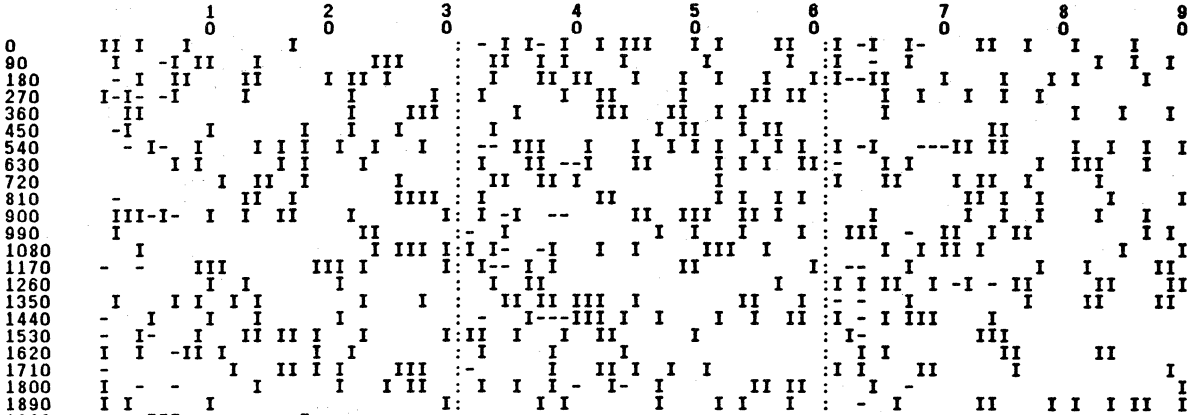
figure 2A

figure 2B



```
I M A G E   S T A T I S T I C S  (B.01.04)
      WED, JAN 26, 1983,  2:44 AM

     DATA BASE: TECUDB          DATA SET (AUTOMATIC): ACCT-SUF-INDEX#2
      CAPACITY: 2003                    # OF ENTRIES: 522
  ENTRY LENGTH: 2                    BLOCKING FACTOR: 30
  % SPACE USED: 26                      # OF PATHS: 2
      ITEM KEY: ACCOUNT-SUFFIX   , I2
```



```
 NUMBER OF SYNONYMS PER PRIMARY.
396      PRIMARIES WITH 0     SYNONYMS
60       PRIMARIES WITH 1     SYNONYMS
2        PRIMARIES WITH 2     SYNONYMS

     458    PRIMARY ENTRIES
     64     SECONDARY ENTRIES

 AVERAGE SYNONYM CHAIN LENGTH =        .14

 PATH  DETAIL SET        ITEM NAME        SORTED BY    CHAIN: MIN   AVE   MAX
   1   CRED-PAY-FILE     !ACCOUNT-SUFFIX                     0     0     0
   2   CHECK-HOLD-FILE   !ACCOUNT-SUFFIX                     1     1     2
```

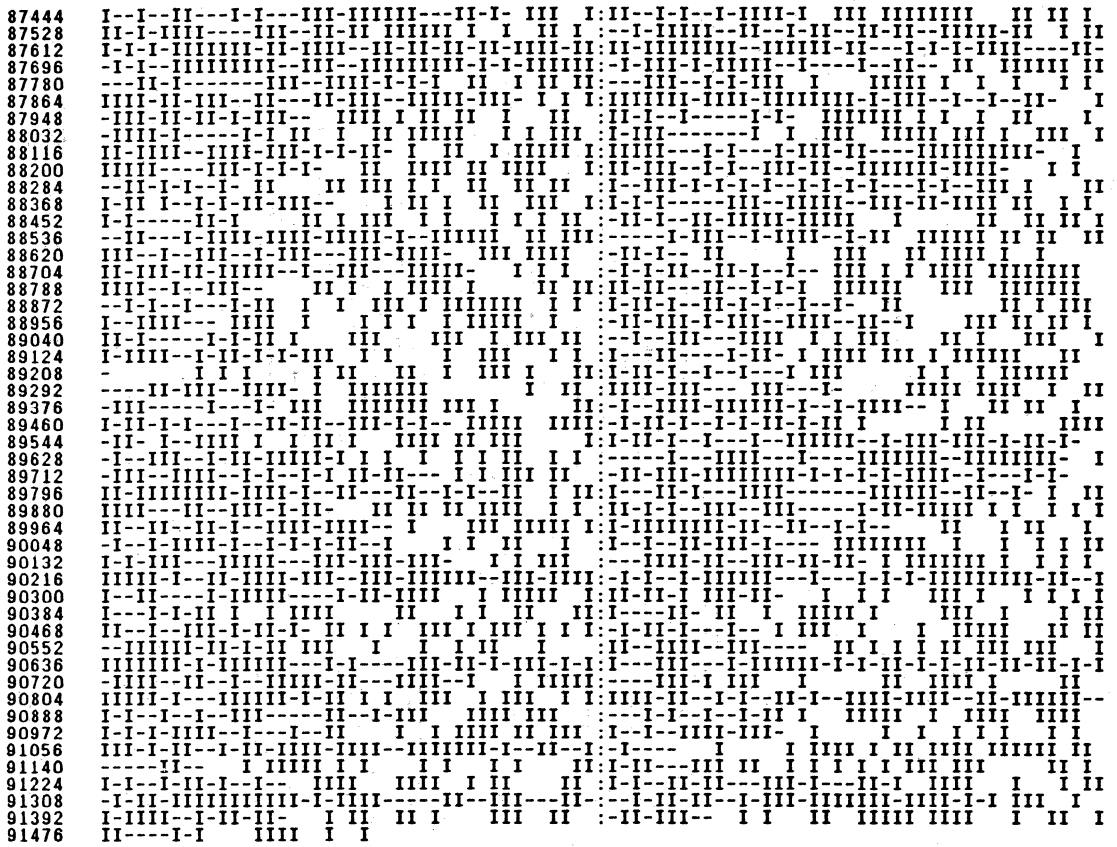
figure 3

```
I M A G E   S T A T I S T I C S  (B.01.04)
      WED, JAN 26, 1983,  2:44 AM

     DATA BASE: TECUDB          DATA SET (AUTOMATIC): SSN-INDEX
      CAPACITY: 91499                 # OF ENTRIES: 72490
  ENTRY LENGTH: 2                  BLOCKING FACTOR: 42
  % SPACE USED: 79                      # OF PATHS: 1
      ITEM KEY: SH-SSN          , I2

                1         2         3         4         5         6         7         8
                0         0         0         0         0         0         0         0
0       -I--IIIIII -I--II-IIIIIIIIII--II-I--I-II-:I---I--II-III-II--II-I-II I    IIII  I
84      III---II-II-IIIIIII-III---I---I-I-- I I I:--I--I---III- III   I II   I III II I  I
168     III-I--III--IIIIII--I-II--II--I--II I   II:I-II-II---IIII-I-I    IIII II  I I    II I
252     I-I--I---I-I--II-I IIIII     I I IIIII II :IIII-IIIIII----I-II------I-II-I-----I--   I
336     -III---II-I--I--II-  II III II II     II I:II-I---III-I-IIII--I-I-I-II  III IIII  III
420     I--I--III--III---    I IIII I I III  I  I :II----I---III III III I  IIII  I   I  III
504     IIIIIIII-I------IIIIII--II-I-II-III-I  I :III-IIII---I-IIIII-IIII-II---I--II  I II I
588     ---I--I-I--II      III   I II        III :-II-IIII--II--   I    IIIIII  III II
672     I----I--I--   III   III  II III II  II I I:---IIII-III---II  I   I III   I  II II III
756     IIIII--II---IIIII-IIII-IIII--II-   I I  I:I-I-III--III--II--I-I-I I   II II   I I
840     --III-II-I---I--I--II I I III  II  III III:---IIIIIIIII-I--II-II---   I I II II I  II
924     IIIIIII-----II-III--II--II--- I II       I:-IIIII--III-I-III-II-II   I I  I  IIII III
1008    II--II-II------II  III III I II II IIII II:IIIIIII-II-II---II--I-III-I-   III  I III
1092    -II-I-IIII-I---IIII- I- I  II   II IIIIIII:IIII-I-I-III-I-----II I  IIIII I II  I
1176    I-II--I-I-I---     II II I I    I II I III:--I---I-IIIIIII-I--II-II-IIIII-III--  I I
1260    --II-I--I-I-II-I--II I I   I I   II   III:--II-III-IIIII-I-III-IIII   IIIII II   II
1344    --IIII-I-I---II--I-I--I-I-III-IIIIII-III-I:---I--IIIII-I---I III  I  I III I III  III
1428    I-I-III-----II-III--I--II   IIIIII  III   :I-I--III-I-IIII-I-II-I-II-IIII- II II IIII
1512    --I-II-III-II----II-  I      IIIIIII  I   :-II---III----II---    II I II  II I I II I
1596    --I-I--IIIII--I--I-I--  II   III II  IIIII:II---II--I-IIIII--I-I--IIII-II-IIII I   I
1680    --I----I-II--I  I   I II    I III  I I I I:II--II-II-III-IIIII-I---III-II-III-II II
1764    IIIII-I----I-II-II I   I I     II     I   :I--I--III---II-III-II--I-- II II   IIII
1848    I--II-II--IIIII--III-III--IIII-IIIII- I  I:-I--II-  IIIIIII I  II II   II I I   I  I
1932    --II-II--IIIII-IIIIIII-I--- IIII I  I II  :I-II---IIIIIIIIIII-IIII-I-I--II-I--I-I I I
2016    -IIII--II-III-IIII-II--II-I--  I    IIIII:--I--IIIII-  I II II III I I  I  I    II
2100    -I--I-----  III   I   I I I   II    I I III:---I-II-III-I-I-I II I IIII   I  II  I I
2184    I-I--I--III--IIIII--II-IIIII-  I  I II I  I:III-IIII-IIII-II--II-IIII-IIIIII--I-II---
2268    -I--I--IIIII-I--II-  III  I  III  I    I I:I--II--I--I II   II                II   IIII
2352    I-I--III--I--II-I   III II  II IIII   II I:--II-IIII-----III-I-I-III II I III II
2436    II-III---III-I-I-I---I I   I III I III II :-IIII-II---II-II-  IIII III  IIIIII I  II
2520    I--I--I---IIIII---    II I II    I I   II I:II-----I-I IIII  I    III   I   II I II
2604    -I---I---III--  I  III    III III III      :-II-I---   IIII     III II II   II     I I
2688    ----I---II   I I II IIIIII    I  III  II I:--II----I-I--I--  IIIIII  I   II  I III
2772    -I--I-I----I-I II III  II     I I I I  III:---I-I--IIII--  I II  II II I II   II I II
2856    ---I-IIIIII-II--III IIII I I I IIIII III  :-II---II-- III  I IIIII III I IIII  II III
2940    I-II-I--III--II-III- I I III I IIII  II II:I--II--I--IIII-I-I-I  IIIII   I I II  II
3024    -I-I  I II  II I    I II    II  III II  I:I-I---- III  II    IIII   I II I  I  III
3108    ----  II    I   I  IIII I  I  I III II   :I-IIII--I--I-IIII-III--IIIII-I  III  III I
3192    IIIIIII-II-I-II--I-I--I--I- I  IIIIII III :I----IIIIIII-I    II    III II IIIIII I I
3276    --I-I--    II  I I  IIIII  II II I II I II:-IIIIII--I-I-II-I-I-II--I-IIII---I   III
3360    III----IIIIIIII---II-II-I-I-IIII-I-IIII- I:I-II--I---II-- III IIIII II I I II  IIIIII
3444    -II-IIII-I-I--I--I     I       II I      :--I-IIIIIII--II--I-  II IIII I I    III
3528    -----III-----IIIII-- IIIIIII I I III I  II:I---II-I-IIII-I- I    IIII II II III    I
3612    -IIIIIIIII-----III-I-I--I-I-III-IIII II II:IIIII-II------I       I  I IIII
3696    --I--I-II-III-IIII III I II I    I  II  I:-------I--  II    IIII I   II   II  I  I I
3780    -I---I--I- III     I  II  III  IIII I    I:-IIIIII-II-II--I---I-II-I--I  II  IIIIIII
3864    II--I-II-----I III  I I I   II  I  I  III :I--I--I-I---II- I I III I I III   IIII I
3948    II----I-  II   I    II  III II   IIII     :IIIII--I---I- III      I   II II      II
4032    -----I-I--- I  II IIII  IIIII II I   I  II:III--I- I   II II  I  II    II I I I I  I
4116    -I-III-I-I--  I II IIII II II II I II  II:I----I-- II    I   I  IIIIII IIIIII I    I
4200    -IIIII-III---I-III-II I          I  III  :-I-I--II---IIII---  III    II  I  I  II II
```

Figure 4A



```
 NUMBER OF SYNONYMS PER PRIMARY.
32894    PRIMARIES WITH 0     SYNONYMS
13016    PRIMARIES WITH 1     SYNONYMS
3454     PRIMARIES WITH 2     SYNONYMS
656      PRIMARIES WITH 3     SYNONYMS
107      PRIMARIES WITH 4     SYNONYMS
6        PRIMARIES WITH 5     SYNONYMS
1        PRIMARIES WITH 6     SYNONYMS

     50134  PRIMARY ENTRIES
     22358  SECONDARY ENTRIES

 AVERAGE SYNONYM CHAIN LENGTH =        .45

 PATH  DETAIL SET        ITEM NAME        SORTED BY     CHAIN: MIN   AVE   MAX
   1    ACCT-SSIN-FILE    SH-SSN                                1     1     158
```

figure 4B

```
I M A G E   S T A T I S T I C S
      TUE, OCT 28, 1980,  3:08 PM

    DATA BASE: PPAYDB          DATA SET (DETAIL   ): TIME-TRANS
     CAPACITY: 5000                  # OF ENTRIES: 178
 ENTRY LENGTH: 45                BLOCKING FACTOR: 10
 % SPACE USED: 3                      # OF PATHS: 1
  CURRENT EOD: 2725                   % DELETED: 93
```

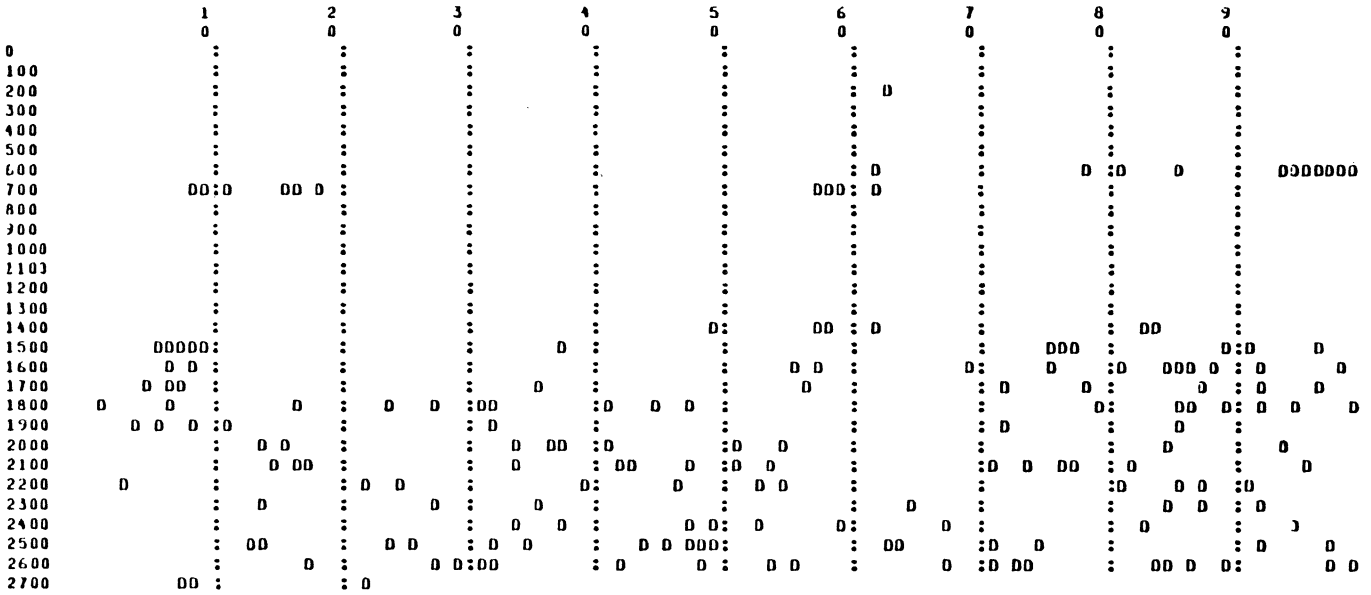


figure 5

```
 NOTE: WHEN INTERPRETING THE DELETE CHAIN STATS,
       REMEMBER THAT THE DELETE CHAIN CAN BOUNCE
       AROUND.  THE NUMBER OF BLOCKS READ WILL PROBABLY
       BE GREATER THAN THE ACTUAL NUMBER OF BLOCKS
       WITH FREE SPACE IN THEM.

   NUMBER OF RECORDS IN DELETE CHAIN: 2547
   NUMBER OF BLOCKS READ WHILE FOLLOWING DELETE CHAIN: 1186
   AVERAGE NUMBER OF FREE RECORDS PER BLOCK READ:      2.1476

   NUMBER OF BLOCKS WITH FREE SPACE: 273
   AVERAGE NUMBER OF FREE RECORDS PER BLOCK:        9.33
```

EFFICIENCY AND OPTIMIZATION OF VPLUS APPLICATIONS FROM AN ERGONOMIC PERSPECTIVE

KAREN BIGLARDERI
NANCY OFSLAGER

HEWLETT PACKARD

EFFICIENCY AND OPTIMIZATION OF VPLUS APPLICATIONS

* DESIGN CONSIDERATIONS FOR DIVERSE ENVIRONMENTS
* OPTIMIZING EXISTING APPLICATIONS
* PROGRAMMATIC INTERFACES
* FORM DESIGN

There are many aspects of man/machine interface that can be considered when designing new systems or optimizing existing applications. The primary criteria should certainly be practicality, speed, and simplicity whenever possible. This paper will point out various techniques that can be implemented by the user when designing or upgrading applications using VPLUS.

DESIGN CONSIDERATIONS FOR DIVERSE ENVIRONMENTS

Environments in which VPLUS is used vary greatly from shop to shop. Many companies use VPLUS only in their data entry department for quick entry of masses of data. Other shops, however, use VPLUS for on-line inquiry and/or update of data bases. The system analyst/designer must consider who will be using the application and what their needs are.

Let's define several different environments and the various factors that should be considered when analyzing and designing software.

Data entry clerks want fast response time and few screen changes. A VPLUS application which changes screens several times per transaction affects their concentration and lessens their productivity. They do not tend to need lengthy directions as their work is very repetitive.

On-line inquiry and update systems are usually used by non-technical people such as order administrators and inventory control clerks. People who are not familiar with computer based systems will gladly accept screens of directions, hints and extensive field labels. They will want exact error messages, catchy field enhancements and will be impressed by meticulous screen designs. The problem with this type of design is that after the first few months when everyone has become quite proficient at using the application, resentment begins to mount at the seemingly endless barrage of messages and questions. Time becomes more important than tutoring.

One HP customer, a large insurance company, designed a system which was to be all things to all people. They wanted a centralized data base that can be accessed by offices in different locations. For example, the computer would be located in San Jose and accessed by the Palo Alto and Livermore offices. If a customer walked in from the street and wanted information concerning his policy, it should be obtained "instantaneously". Data should also be able to be added and modified locally as necessary. So we have a

case of high volume, on-line inquiry/entry/update processing that demanded satisfactory performance (response time). The problem was that it took thirty seconds to display a form. This is obviously unacceptable response time for a busy on-line inquiry/update application. Much of their problem was due to the fact that the programs did not take advantage of many of the VPLUS optimizations and also neglected elementary considerations for the type of environment in which they operated, one which necessitated manipulating tremendous masses of data repetitively and frequently.

The design may have been fine for an environment which needed only occasional access to and minimal information from a data base, but in this case it was very inefficient.

Important design considerations in this case should have been:

* Use of form families to greatly reduce datacomm overhead and screen paint time

* Keeping to a minimum the number of times screens are changed during transaction processing

* Not using process handling; Process handling causes VPLUS to override VHOWFORM optimizations and prohibits the use of forms cache

* Use of HP2624B's (thereby making available forms cache and local edits

* Understanding that use of MTS increases response time

* Understanding that use of MTS decreases available terminal memory, which can be significant if forms cache is being used

* Using SHOWCONTROL to avoid repainting the screen, enhancements and data if this information is unchanged

Another environment that is quite common is the order processing/ inventory control type of environment. In this case, the application usually is used for some on-line data base inquiry ("How many parts do we have on backorder?"), some on-line data base maintenance (adjusting quantities of available parts), and some data entry (adding a new order to a sales data base). There is not a tremendous amount of data to be handled or as great a need for speed as in the previous example.

When designing for this environment, important considerations should be:

* Infrequent screen changes per transaction

* Menu driven screens for user friendliness and ease of structured programming techniques

* Comprehensive error messages

* Initialize fields whenever possible to reduce typing by user

* Use of local edits when possible

OPTIMIZING EXISTING APPLICATIONS

Often, systems are not designed to perform at peak efficiency. However, there are some techniques that can be used to optimize these existing applications.

Local form storage (LFS), a VPLUS feature designed to make use of special terminal features, allows forms to be kept in and displayed from terminal memory. This feature is available on the HP2626A and the HP2624B terminals. Although it is implemented differently on the two terminals, a local form storage application will provide two obvious advantages over a non-local form storage application. The first feature of LFS that will help optimize existing applications is the reduction in data communication transmission. In a well designed system, a form will be transmitted to the terminal as few times as possible, preferably only once. The "FORM" refers to the empty form design created in Formspec. Initial values, enhancements, window messages, or function key labels are not stored locally. There are at least as many terminal writes for a form transfer in an LFS application as in a non LFS application. Where the savings occurs is in the number of characters transmitted over datacomm. As the form will be resident in and accessed from terminal memory, subsequent access of the form will trigger an escape sequence rather than the entire screen design. Since a large VPLUS form can contain thousands of characters, the savings in datacomm transmission can be significant.

The HP2626 can hold a maximum of four forms and the HP2624B can hold up to 255. When implementing LFS on the HP2626, the systems designer should design the application to revolve around four (or fewer) forms. For example, if an application has twelve forms, it should be designed such that each transaction type will need no more than one set of four forms. When a particular type of transaction is requested, load the

four forms, and require that all transactions of this type be performed before another type is selected. When the new transaction type is selected, load the next set of forms that are necessary.

There are currently two methods of loading forms, automatically (using the look ahead option) or manually (using the VLOADFORMS intrinsic). Using the look ahead option is by far the easiest because VPLUS, not the user, is determining which forms should be loaded or unloaded. When word 32 (LOOK'AHEAD) of the comarea is zero and word 39 (FORM'STORE'SIZE) is greater than zero, look ahead pre-loading of forms will occur. The next form, as specified in Formspec, is downloaded during VREADFIELDS.

There will be available in Q-MIT a third way to load forms. Setting bit 9 (adding 64) to the SHOWCONTROL word in the COMAREA will enable preloading of forms during VSHOWFORM. If bit 9 is set when VSHOWFORM is called to display a form not in the currently loaded series of forms, the form will be loaded, then displayed. If bit 9 is not set, the form will be displayed, but not loaded.

Forms are purged to make room for new forms on a least recently used basis. This method will load a single form at a time. Multiple forms may be loaded by using the VLOADFORMS intrinsic. When loading and unloading forms manually, the user must be aware that he is responsible for monitoring the forms.

The second advantage which is realized when using local form storage is an aesthetic one. Users will not have to wait for and watch the form being painted on their terminal. A locally stored form flashes up on the screen in its entirety instead of painting line by line.

Because of the way in which it is implemented, the display time varies slightly on the two different terminal types. On the HP2626 the screen display is almost instantaneous. This is because this terminal uses the workspace method, in which displaying a form is simply a matter of closing one workspace and opening another. The form is stored in displayable format and showing it is only a matter of switching to an active workspace.

On the HP2624B display time is somewhat greater because the form must be copied from cache memory to the display area. During this operation, the terminal executes the escape sequences contained in the compressed form and creates the displayable form. The display video, which is turned off during the copy, is then turned back on.

The advantage of the HP2624B's method is that there is always a clean copy of the form displayed. On the HP2626, any error enhancements, field type changes, or modifications are made directly on the stored original.

For more extensive information regarding local form storage and its implementation, please consult the article in the Communicator Issue #30 by Ron Harnar.

Another feature of VPLUS which can be used to optimize existing applications is the SHOWCONTROL word. A default value of 0 in SHOWCONTROL prevents a screen from being redisplayed if no information has changed. This optimization can be overridden by setting different bits in SHOWCONTROL word in order to force the display of unchanged forms, data and enhancements, and the window display line.

Designers should carefully consider the overhead involved in forcing the display of unchanged data. For example, let us suppose that a form the size of the Formspec main menu is repeated in place, and SHOWCONTROL is altered to force the redisplay of the form each time. Rather than transmitting the screen design via data communication and painting the screen once for the application, the screen design will be transmitted, and redisplayed every time the enter key is pressed. As the compiled screen design of the Formspec main menu is approximately 660 bytes, this amount of data would be transmitted to, and displayed on the terminal every time VSHOWFORM is called. This can be very costly in terms of data communication overhead, especially on a heavily loaded system.

PROGRAMMATIC INTERFACES:

The Autoread feature

The AUTOREAD feature allows data to be transmitted from the screen buffer to data buffer if a call to VREADFIELDS has been terminated by a function key instead of the ENTER key. Hitting a function key does not now and never has transmitted data to the data buffer. By enabling the Autoread feature, this can be accomplished. Follow these steps:

Following a call to VREADFIELDS,

1. Determine that a function key has been pressed by interrogating COM-LASTKEY (word 6 in the COMAREA) for a non-zero value. It will contain the value (1-8) of the key that was pressed or zero if the ENTER key was pressed.

2. Set bit 14 in TERM'OPTIONS (word 56 in the

COMAREA). If programming in COBOL, use the ADD instruction to add 2 to the word. Do not use MOVE as this will destroy the bit configuration of bits 0-13 (part of which contain the FCONTROL 31 handshake protocol) and cause an error 151 "EXPECTED DC2 FROM THE TERMINAL MISSING".

3. Call VREADFIELDS. With bit 14 set, data will be transmitted from the screen buffer to the data buffer.

4. Clear bit 14. COBOL programmers should subtract 2 from word 56.

Function key labels

Function key labeling is available on the HP262X family of terminals. Designers should remember that function key labeling is not supported with process handling because of the method VPLUS uses to monitor changed labels. A bit map is kept in a table which is inaccessible to the user. During process handling labels can be altered without altering the corresponding bits of the appropriate bit map. Thus when control is returned to the father process, incorrect labels (ie. the other process' labels) may be displayed. It is possible to rectify this situation upon return to the father process by performing two calls to VSETKEYLABELS. The first call to VSETKEYLABELS should set the labels to some dummy values, blanks for example. The proceeding call to VSETKEYLABELS should set the desired label values.

The function key label option can be used on the HP264X terminals by designing the labels into the form and using the VGETKEYLABELS intrinsic programmatically to retrieve the labels from the forms file. They will be returned to a buffer and can then be moved to the screen buffer and displayed using VPUTBUFFER and VSHOWFORM.

Break/Nobreak

If a user wishes to disable the break option, a UDC may be set up which specifies NOBREAK. As the BREAK key can easily be mistaken for the ENTER key on HP264X terminals, this option may be desired for non-technical people who would not be able to recover from hitting the BREAK key in block mode. The UDC would look something like this:

```
*******
PAYROLL RUN
RUN PAYROLL.PAY.MAINT
OPTION NOBREAK
*******
```

FORM DESIGN

VPLUS/3000 can be used with most HP264X, HP262X, HP307X and HP2382 terminals. As VPLUS uses many features of the various terminals, the type of terminal(s) on which the application will be run should be of major concern to the designer.

The HP264X terminal family is the terminal default - you do not need to access FORMSPEC's Terminal Selection Menu to select this terminal type. When designing forms to run on these terminals, the following constraints should be considered:

* Do not use column 79 if the form is to be part of a form family

* Do not use column 80 if the form may have another form appended to it or be part of a form family.

* The security display enhancement is not available on the HP264X.

The security display enhancement suppresses character printing by turning off the echo. It is useful for password fields. Even though the security display enhancement is not available for HP264X terminals, it is possible to turn off the echo for password fields by shifting to an alternate character set that the terminal does not support. For instance, in Formspec set the field to escape sequences using display functions as follows:

******Processing Specifications*****

INIT
SET TO "esc)ANc."

(escape right paren capital A control N dot)
This would change to alternate character set A. It is automatically terminated when the user goes to the next line, so you would probably want this field on a line by itself.

VPLUS runs on the HP262X terminals without any special action on the part of the user. However, to take advantage of available options, the HP262X Family must be specified on the Terminal Selection Menu.

LOCAL FORM STORAGE, as previously discussed, is one of the special features that exists on the HP2626 and HP2624B terminals. The use of LFS can greatly increase

the efficiency of a system, and if the supported terminals are available, they should be utilized. An application which will be used on both terminals supporting LFS and terminals not supporting LFS can easily be designed, as the LFS intrinsics are not executed if the terminal is not one which supports the feature.

LOCAL EDITS are another terminal feature available on the HP2624 terminals. They are edits which are performed locally within the terminal as keys are pressed by the user. These differ from program edits or VPLUS edits, which are performed after the user presses the ENTER key and control is passed back to the application. Local edits are specified in the CONFIG phase of the processing specification section of the field menu in FORMSPEC. For example if the field must be alphabetic, it could be set up in FORMSPEC like as follows:

```
****Processing Specifications***

CONFIG
LOCALEDITS ALPHABETIC
```

The escape sequences necessary to perform these edits are compiled into the screen design, and are loaded into the terminal by VSHOWFORM when the form is displayed. If the terminal is not capable of performing the local edits, VPLUS ignores the escape sequences without any adverse effects.

The SECURITY DISPLAY ENHANCEMENT is available on HP2626 and HP2624 terminals. As previously mentioned, it turns off the echo so characters do not show when typed in. It is enabled by placing an "S" in the enhancement box on the Field Menu in FORMSPEC. The field to be secured must be delimited by visible brackets.

The LINE DRAWING character set, a terminal option, and the "draw line" function keys on the HP2626 can be used to create very concise, pleasing forms. The form designer may want to consider using line drawing when the users involved are accustomed to using accounting sheets. It is easier to train new users if the online applications closely resemble the old form of paperwork.

PERFORMANCE CHARACTERISTICS OF HP/DSN (DS-3000)

by Mel Brawn, Senior Systems Engineer
IND Division, Hewlett Packard Company

ABSTRACT

DS-3000 is being used by a large number of 3000 users. The applications of various users vary, and the performance of these applications also vary. There are certain basic activities such as file access, F Intrinsics (FREAD,FWRITE, etc.), PTOP, Remote Data Base Activity, and NFT.

The performance characteristics of these separate activities can be analyzed. The overall network performance then depends, not only on the activity being used, but also on the simultaneous activities of other users. This paper evaluates the basic activity building blocks, then predicts the affect on performance of multiple users. Certain guide lines are established for optimizing performance for an overall network. New techniques utilizing the Inter-Process Communication (MSG Files) are discussed. The techniques discussed are useful in evaluating existing applications as well as designing new ones.

INTRODUCTION

The Hewlett Packard Distributed Systems Network (DSN/DS) utilizes the DS-3000 product. It is currently specified in a large percentage of HP 3000 computer systems. The purpose of this paper is to provide the information required to understand and properly utilize the DS system in an applications environment.

DS/3000 was introduced in 1977. Since that time enhancements and modifications have been made to improve the effectiveness and performance characteristics of the product. At the present time DS is available in either the BISYNC version, or in the X.25 version. This paper will deal primarily with the BISYNC implementation. In a companion paper the characteristics of the X.25 implementation will be discussed (1).

DS is easy to use. Some existing activities can be executed using DS between adjacent systems with no changes to programs or files. The ease of its use is sometimes misleading. Performance may be disappointing if no consideration is given to the application. The details contained in this paper will be useful in planning new network activities, or in analyzing existing network applications.

The data presented covers the Intelligent Network Processor (INP) in conjunction with the series 30/33/40/44/and 64 computers. Although no specific data is provided using the HSI or SSLC assemblies on the Series II and III computer family the same applications techniques are relevant.

The DSN/DS capabilities include Remote Commands, Virtual Terminal, File Transfer, Remote Data Base Access, Program to Program communications, and file access including the use of IPC files on remote Systems.

APPLICATIONS ENVIRONMENT

The DSN/DS supports multiple users over the common communications link. A user environment consists of a local session and a remote session on the adjacent node. For multiple node networks a remote session is required on each successive node. There may be a number of simultaneous users on any given DS line.

The users may access DS directly with PTOP activities, or indirectly through the File system, IMAGE subsystem, Command Intrepreter, etc. High level DS provides the mechanism for multiplexing various users over the line. On the remote system applications activities are handled through the remote user session, such as file access and remote commands.

PROTOCOL

The BISYNC type protocol when used with DS/3000 (i.e., non X.25) provides for conversational acknowledgements. Each request across the line at the users level requires a reply. These requests and replies are paired, although they need not be consecutive. The communications link may be full duplex or half duplex, but the transmissions from both directions are non-simultaneous.

DS is activated by the DSCONTROL console command. Either end may initiate the actual connection. This initial connection involves a dialogue which determines line buffer size, compression capability, block numbering sequencing, multiple packet and exclusive user activity. The first connection is made on behalf of the first DS user. Thereafter additional users can use the line, but no further changes can be made in its characteristics, without shutting and reopening the line via the DSCONTROL command.

When no line activity has taken place in a while (on a dial up modem line) the line reverts to the control mode. In control mode a line read exists from both ends. Therefore a transmission request results in a bid for the line. The read is terminated, an ENQ requests the line, and an ACK0 reply allows the data to be transmitted. The line then is reversed for subsequent activity using BISYNC conversational acknowledgements. When no outgoing traffic is required for a while DS

turns the line around to facilitate possible activity from the other end. The line turn around is accomplished with a 'null' message. When both ends have no traffic, the 'null' 'null' exchange is terminated with an EOT and the line reverts to the quiescent control state.

A modem line in the quiescent state has 'line-keep-alives' approximately every 20 seconds. This is an ENQ - ACK0 - NULL - EOT exchange.

There are three types of line activities in the protocol sense. The first include BISYNC control exchanges. These are typically one or two control characters such as ENQ, ACK0, EOT, etc. The second type includes DS requests and replies. They look like BISYNC transparent blocks, with a DLE STX at the beginning of the block and a DLE ETX BCC BCC at the end of the block. The DS data will contain a fixed header consisting of 16 characters. This header provides information on the nature of the block, who it is from and who it is to, length of the block, etc. There is then an optional appendage. The length is variable. It contains information concerning the actual intrinsic being executed, such as status information, record number, successful completion, etc. For PTOP operations a tag field of 40 characters is provided. Then an optional data field contains the user information.

The third type is the 'null' exchange. It consists of: DLE STX - 1 seq DLE ETX BCC BCC. It appears like a normal transparent block at the communications link level, but it has no DS Header.

Error recovery techniques follow the normal BISYNC techniques. A message is NAK'd if the BCC CRC-16 check fails. If a message is lost due to failure to achieve sync between the modems, or line hits that result in failure to recognize the data block then the BISYNC time outs provide for recovery. Detailed descriptions of error recovery exceed the purpose of the paper.

USER QUEUING AND DATA FLOW

A user of the DS line may do some programmatic activity such as a file read. The file system services this request. It prepares the appropriate DS header and appendage. Refer to Figure One. The request is then processed by the high level DS code. The request to the line results in an IOQ on the IODS0 dit. The DS I/O Monitor services this request and passes it on to the DSMON process. DSMON transfers the user data to its extra data segment. It performs the breaking into continuation records when required, as well as data compression if required. The data is then sent to the low level drivers where the BISYNC control characters are added to the buffer, and the CRC-16 check characters are determined. The INP transmits the data across the line.

On the remote end the low level drivers receive the block. It verifies the BCC character check and passes the block to DSMON. DSMON services the block and passes it on to the DS I/O Monitor. The

DS I/O Monitor passes it to the appropriate user level subsystem, in this case the File System. If another user has an outgoing request or reply this traffic is handled by DSMON and it is transmitted over the line. The File system services the request and prepares the reply block. This reply is serviced in turn by DSIOM, DSMON, and the lower level drivers. The data is transmitted over the line. When it comes back to the local system the DS reply is checked for BCC check and sent to DSMON. DSMON then returns it to the DSIOM. DSIOM returns it to the user activity, the File System. The File system completes the user intrinsic and returns the data to the user stack. This completes the request and reply. The user program is then free to continue its execution.

The important aspects of this transfer include the automatic features such as continuation records, compression when requested, multiplexing with other active users, and transparent data transfer.

USER APPLICATIONS

A wide range of user activities can be supported over the DS line. We shall try to address them individually, and then consider these simultaneous activities on the line.

The user must have a remote session. This can be initiated interactively with the appropriate commands, programmatically with appropriate commands, or with UDC commands. In most cases one goal is to make the overall application as friendly to the user as possible.

The user may use remote commands and virtual terminal activities without worrying about the details. There will be some degradation in performance compared with local terminal activity on the ADCC or ATP. These type of activities are generally straight forward and provide little opportunity for performance optimizing.

File system intrinsics between remote nodes may offer a wide range of performance characteristics depending on the application design. In general the data should be processed over the DS line in large effective blocks. The total throughput through a given line depends largely on the nature of the applications activities. Where the user can block the data up to larger effective sizes the total throughput can normally be improved. Since the system can break the data into whatever size blocks are required for the configured line buffer size the user reads/writes can be large. The DS subsystem will break the data into continuation records as necessary. The total overhead will be reduced compared with a larger number of activities consisting of smaller user blocks.

Remote Data Base Access can be done transparently using File Equations. The user needs a remote session. Then the file equation indicates the communications link. The performance

characteristics of RDBA compared with local data base access indicates an increase in CPU load, and elapsed time. The data from previous studies indicated an average increase in wall time of up to about a factor of four for RDBA compared with local data base access. There is also an increase in the total CPU resources required of 3.5 or 4 to one, with one to two ratio on the local and remote systems. This means a series of Data Base activities that might require 10 minutes on a local system might require 35 to 40 minutes when the data base is remote. This assumes that there is negligible contention for the line. A series of activities that require 1000 CPU seconds for execution locally might require 1000 to 1200 locally and 2000 to 2500 remotely when accessing a remote data base. These studies included a wide range of Image intrinsics with rather heavy use of random access. The design of the application using PTOP was more effective. In the PTOP application summary type data was transferred between the processes. The data base access was local on both machines. At the present time these types of intensive RDBA are also being implemented using the Message file techniques. Although the details are beyond the purpose of this paper a user will want to consider PTOP or IPC techniques for an application making heavy use of remote Data Base Access.

File transfer can be easily done using the DSCOPY Network File Transfer Program. The program provides much more efficient file transfers than FCOPY. DSCOPY uses a 4000 byte buffer. It fills it up as much as possible utilizing the file blocking factor. The blocking factor is still very important. A blocking factor of one or two greatly increases the CPU load, decreases the throughput, and increases the disc accesses per second. The optimal blocking factor depends on record size, but 16 seems to be a good number for typical source files. 16 * 80 = 1280, so the buffer holds three of these reads. The default for FCOPY is one record at a time, so for a given file a much larger number of DS requests must be processed.

DSMAIL also uses 4000 byte buffers in a PTOP environment. Its characteristics are very similiar to DSCOPY.

The use of IPC files has greatly expanded the range of applications for network activities. In general two processes wishing to communicate must open two IPC paths. A local process opens a local IPC file as a reader, and opens a remote IPC file as a writer. The remote process opens the remote IPC file as a reader and the local IPC file as a writer. Thus there exists two one way paths between the two processes. This technique works in the same manner whether on the same machine or on adjacent machines. There are a number of advantages compared with PTOP. The transfers are core to core, except that disc back up is available when the reader is unable to keep up with the writer. There is no master/slave relationship, both processes are equal. The communications paths are bilateral and simultaneous. Message files also provide suspension if there is nothing to process, timed reads, writer ID, IO without wait, interrupt to a procedure, etc.

Applications using IPC file techniques can often reduce the

requirements for each user to have a remote session. Programs can be written to provide rational incoming and outgoing servicing of the DS lines. They tend to reduce memory requirements. Only a single user needs to have the remote session, compared with non-IPC applications in which each user must have the remote session.

Every application design should be verified by the use of the CS Trace capability. There are sometimes surprises when some factors have been overlooked in the design. DS Trace can be activated by the DSCONTROL command. When the trace is deactivated then an analysis can be made using either DSDUMP or CSDUMP. The CSDUMP program does a formatted print of the data that can then be analysed. The DSDUMP program provides a higher level formatted printout. It can also be used interactively to a terminal and greatly enhances the interpretation of the DS activity.

The Compression algorithms have been greatly improved. At the present time we recommend always using compression. The amount of CPU work required to perform the compression is reduced so that it is usually beneficial up to and including 56 kbits/second. Generally user data does contain some redundancy. Source files, for example, may contain up to 40 to 60% redundancy. The compression algorithm follows the SNA specification. Any group of three or more consecutive identical characters can be compressed. Generally there are a great deal of spaces in source or print files. The users data is compressed, but not the Fixed Header or Appendage.

FACTORS WHICH AFFECT PERFORMANCE

There are a multitude of factors that affect the overall performance over a DS network. The instantaneous line speed provides an upper limit. A 4800 bit/second line has an ultimate limit of 600 characters per second. We can never achieve this because the line is not fully utilized, and because there are protocol characters (BISYNC envelope) and DS Header and Appendage. Thus the effective user data will be able to approach 65 to 85% of this figure with proper attention to buffer sizes. The modem line turn around delay can also be significant. Values range up to 150 or 200 milliseconds for half duplex modems. Transmission over a satellite link will experience delays up to 230 to 250 milliseconds for an up and back one way trip. This is especially devastating for small data blocks.

There is a certain amount of CPU crunching that must be done to service each DS request. The speed of the CPU will affect the throughput. If one factors out the time spent on line transmission and line turn around delays, and also removes the affect of disc accesses then the portion of the time spent within the system is almost directly related to the CPU speed of the various computer systems. A 40 or 44 will be faster than a 30 or 33, and the 64 will be faster yet. Applications activity is seldom stand alone, so other CPU load must be considered. Some further information on the nature of the load is necessary to determine overall throughput. If CPU loads are primarily CPU bound then the priority of the various processes will be significant. In systems in which tight memory availability results in

memory management the DS performance may be affected. In many larger systems the other application loads may result in queuing for the disc. In applications that require heavy disc accesses on the system there may be significant affect on those DS applications that also require disc access. DSCOPY transfers with poor blocking factor might be an example.

In DS applications requiring local or remote data base access the queuing for IMAGE disc access may cause large or unpredictable variations in performance.

The mix of simultaneous activities on the DS line will also provide variations in performance. The user queuing on the line is handled by DSIOM. Once a user's IOQ is being serviced no other user request can be serviced until the data is sent and the line is reversed by data or a 'null' reply. If a user's buffer is large (i.e., 8 to 10 kbytes) there may be a large delay before the next user is processed. A mix of activities, for example, a DSCOPY file transfer at the same time a user is doing 200 byte FREAD or FWRITE will result in unequal degradation to these two users. A 4000 byte buffer will be processed for DSCOPY for each 200 byte transfer for the second user. The throughput for the DSCOPY may drop from 3500 to 3000 char/sec in the presence of the second user (assuming 56 kbit line). The second user may drop from 1200 to 300 char/sec in the presence of the DSCOPY transfer. Actual response time or transfer rate may vary widely depending on system load, and activity mix.

The total throughput will generally be higher when multiple users share a DS line. When user activity is queued up DSIOM can immediately pass the new activity to DSMON and the line utilization is high. If no user activity is ready then the DS system waits for a short time to see if activity develops. The timing algorithms were developed to maximize the transfer rates. With only a single user active on the line the time delay before turning the line around with a 'null' increases to 1.2 seconds. When multiple users are active on the line the time decreases to .3 seconds. Thus, in a particular application there may be some delays in turning the line around.

The processing for a user's request takes place at the priority of the user's session. The DSMON and low level driver activity takes place at the DSMON priority. Thus, incoming and outgoing traffic takes precedence over user activity. But the user level code must be executed at both ends to complete DS activity. Depending on the nature of the application and the desired performance the intentional selection of higher or lower priority might be appropriate.

PERFORMANCE RESULTS FOR FILE ACCESS

Figure Two shows the performance of a DS line running at either 56 kbit or 19.2 kbit. The line activity is Fwrites to a message file on the remote. This data is for a Model 40 or 44 on a lightly loaded system. You will notice a significant difference based on record size. The line buffer size is configured for 1024 words. When the user buffer exceeds approximately 1975 bytes a continuation record is required.

This results in a saw tooth effect in the performance. The actual transfer on the line includes the initial request, and its reply, followed by the continuation request, and its reply, etc. Notice at 56 kbit the maximum user transfer rate approaches 4500 to 5000 char/sec. This data is for lightly loaded systems essentially dedicated to the DS activity. In cases in which the system is moderately or heavily loaded the throughput can be degraded to 40 to 60% of this figure.

Figure Three shows the same data presented in time/record form. You will notice the minimum time is approximately 100 milliseconds for a very small record. This would indicate an ultimate maximum rate of 8 or 9 record transfers per second. With a record size of 2000 characters a maximum transfer rate of two transfers per second might be experienced. Again, with moderate or heavily loaded systems these figures may be decreased by 40 to 60%.

Figures Four and Five provide the same information for line speeds of 2400 to 9600 bits/second, with different line turn around delays. You will notice that with lower speed lines the user can approach a higher percentage of the instantaneous line speed. This is because the effect of the system delays is less significant due to the pacing of the slower line. At lower line speeds system delays due to CPU activity or disc activity also cause somewhat smaller impact on performance.

The test set up for these Freads and Fwrites included data that was not compressable. A program buffer was set up so no disc accesses were required on the local end. The data was transmitted to a message file on the remote end. Another process reads the data from the remote message file buffer so again no disc accesses were required. On the local end the CPU load ranged from 10.8% for a record size of 80 bytes to 7% with a buffer size of 8192 characters (for the 56 kbit line speed). On the remote end the CPU load varied from 28% to 5% depending on record size. This included the contribution from both the remote session of the originating user as well as the process on the remote computer that read the data from the message file. With compressable data the CPU load would be slightly higher. The apparent throughput would increase from a negligible improvement with small records to a factor representing the percentage of compressibility for large records. Data that involved disc reads or writes would also impact the performance.

PERFORMANCE RESULTS FOR DSCOPY

Figure Six indicates the thoughput on a series 44 at 56 kbit, and 9600 bit/second with a full duplex line, and at 2400 bit/second with a 150 millisecond turn around delay. You will notice a wide variation in throughput depending on the record blocking. This effect is greater at higher line speeds. This data was taken with no compression. The corresponding figures for a Fortran source file is provided at 56 kbit/sec when using compression. The file exhibited 60% redundancy. The total throughput reached 5580 characters per second with optimal blocking factor. It may require 10 to 15 seconds

to set up the DSCOPY process on two systems, and to initialize the new file. Once this is established the steady state transfer rate raises up to about 8700 char/sec for this example with 60% redundancy. The higher figure is appropriate when using large files, or for the second or subsequent file so that set up time is less significant. This data was taken with no system load. With moderate to heavy system load these figures should be reduced to 40 to 60% of the no load values. The nature of the system load may also impact the file transfer. At 56 kbit approximately 20 disc transfers were required per second with a blocking factor of one. With a blocking factor of 16 the disc access rate drops well below one per second. The optimal blocking factor helps minimize the effect on the system.

The CPU loading varied from about 4.4% with a blocking factor of 16 to 7.4% blkfact=1, with no-compression and line speed of 56 kbits. With compression it ranged from 9.3 to 12.4%. The total amount of work expended for a given file depends on blocking factor and compression but is independent of speed. Thus a slower line speed tends to pace the CPU work and reduce the instantaneous CPU load. The CPU load on the destination end is slightly lower but of the same magnitude.

When transferring with DSCOPY there is set up time to establish the DSCOPY PTOP environment on each system. There is also time required to initialize the file. These figures are included in the overall throughput. Instantaneous transfer rates (i.e., excluding this set up time) may be 10 to 15% higher.

EFFECT OF LINE ERROR CONDITIONS

Figures Seven to Ten provide information concerning the optimum line buffer size as a function of line error rates. Figures Seven and Eight show optimum buffering for transmissions at 4800 bits per second with light and heavy CPU loads. Figures Nine and Ten show the data for transmissions at 2400 bits per second.

The characteristics of line errors does affect DS performance. Normally line error rates are specified in a form such as: 80% of the time line error rates will not exceed 1 part in 10**4. Unfortunately the maximum error rate is not provided. The actual rate depends on the type of line, the quality of the Telephone service, the time of the day, and many other factors. The nature of line errors is usually 'hits' on the line rather than merely random bit errors. These hits may last up to a few milliseconds. Whole sections of a transmission can be lost. Normally leased lines provide better quality, and less variation in error rates than dial up lines. Dial up lines may take a different path each time the circuit is established.

For transmission over a noisy line the optimal buffer size may be as little as 500 to 800 bytes. In general the user can determine the quality of the line with a couple of techniques. The user can listen to the quality of the tone when the carrier is established. If the tone varies or warbles it is appropriate to hang up and re-dial. If the tone sounds good then proceed with the DS activity. Once the

transmission has begun the user can use the :SHOWCOM XX,ERROR command to determine the number of line errors and retransmissions. If the traffic will be lengthly one would not want to continue if there are a high percentage of retransmissions.

As a rule of thumb a buffer size that provides at least 3 seconds of line transmission seems to be good with most dial up lines. For 4800 bit/second operation the full INP buffer of 2048 characters seems to be satisfactory. On slower 2400 bit/second lines this may be excessive. The user is encouraged to use SHOWCOM to determine the statistics for the communications lines normally utilized. Using the full INP buffer limit might be satisfactory unless undue line error conditions suggest a smaller size.

The DS subsystem will retry in order to re-establish satisfactory communications. However, if this fails, and the error retry counter is exhausted, then the line must be closed before reuse. When an irrecoverable error is detected the remote sessions are terminated. The local sessions are marked dirty so that continued activity is not allowed. The user would then close the line, open the line, and re-establish the connection. The application should be designed robustly enough to allow picking up and continuing the activity once the line is re-established.

HP ASSISTANCE FOR DATA COMMUNICATIONS CONSULTING

There are HP Systems Engineers that have been specially trained to handle Data Communication network design and analysis. You are encouraged to contact the HP field office for further information and assistance.

SUMMARY

With a given network configuration there are many things a user can do to optimize the performance of an application. Try to minimize the number of request-reply pairs required by buffering larger amounts of user data for each request. Try to send processed or summary data between nodes rather than raw data when this reduces the data traffic. Be concerned with applications activity that impacts the system, such as the number of disc accesses required. Use properly blocked files for improved performance. On adjacent nodes use IPC or PTOP techniques to improve the performance of remote data base access. Design applications so that activities interrupted by faulty lines can be easily reestablished.

Single processes servicing incoming and outgoing traffic using IPC file techniques may save significant amounts of memory. Simultaneous users may make more effective use of a DS line for total throughput. Applications utilizing remote terminals can be optimized to reduce the number of terminal control and data exchange transactions. The use of DS Trace can verify the actual line traffic. SHOWCOM will show the current activity on the line, and the number of

retransmissions required. User buffer sizes of just under 2000, 4000, etc usually provide the maximum throughput for any line speed.

(1) Carol Hibbard and Mel Brawn/Hewlett Packard
"Selection Criteria for Choosing Bisync or X.25 Protocols for use with DSN/DS"

FIGURE ONE
DATA FLOW

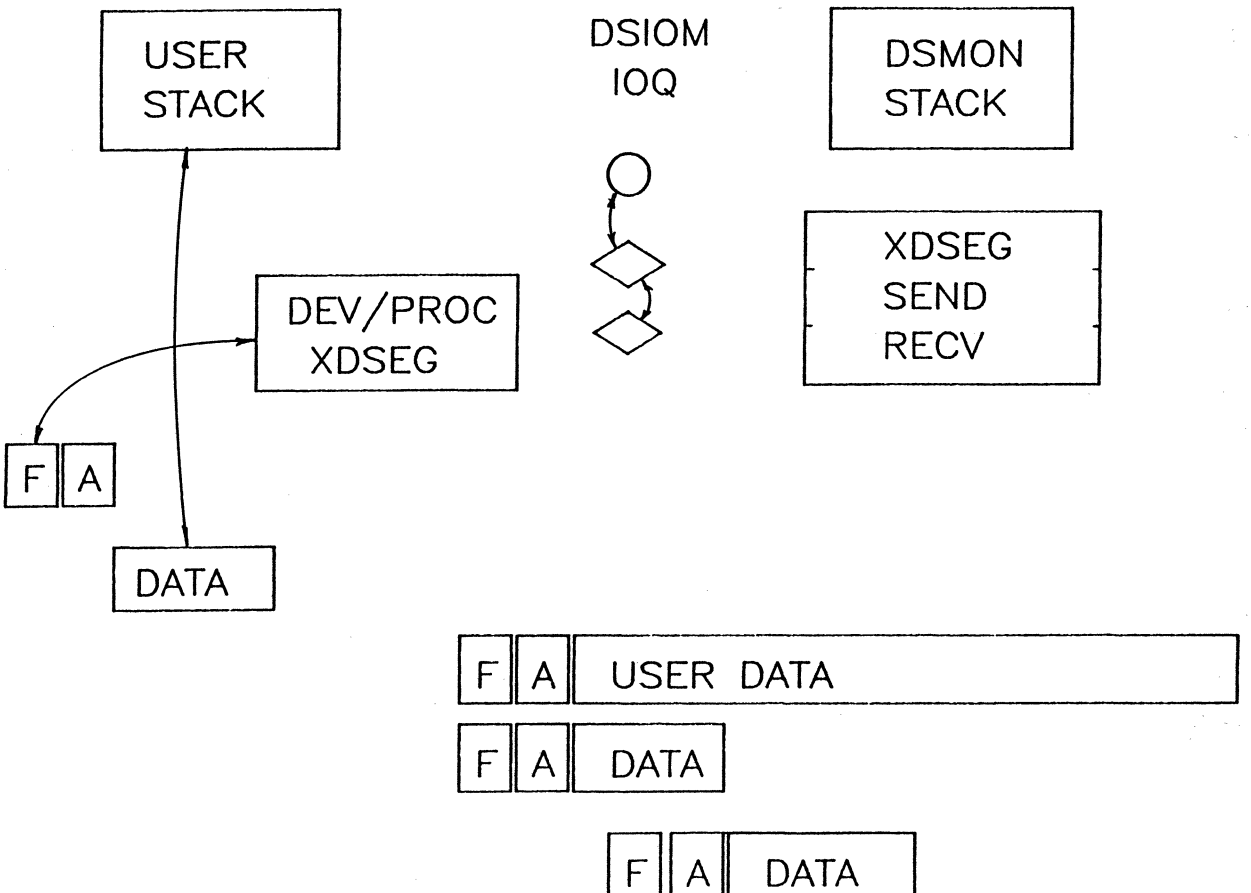


FIGURE 2. RFA PERFORMANCE

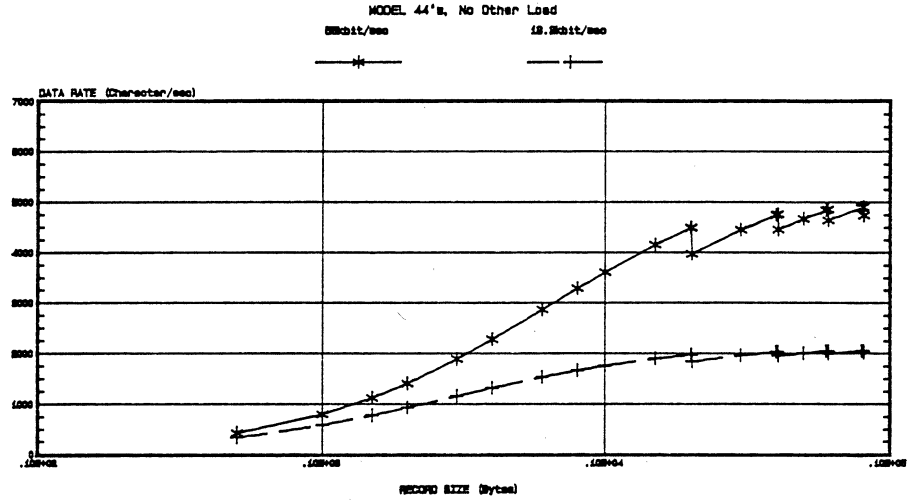


FIGURE 3. RFA TIME/RECORD

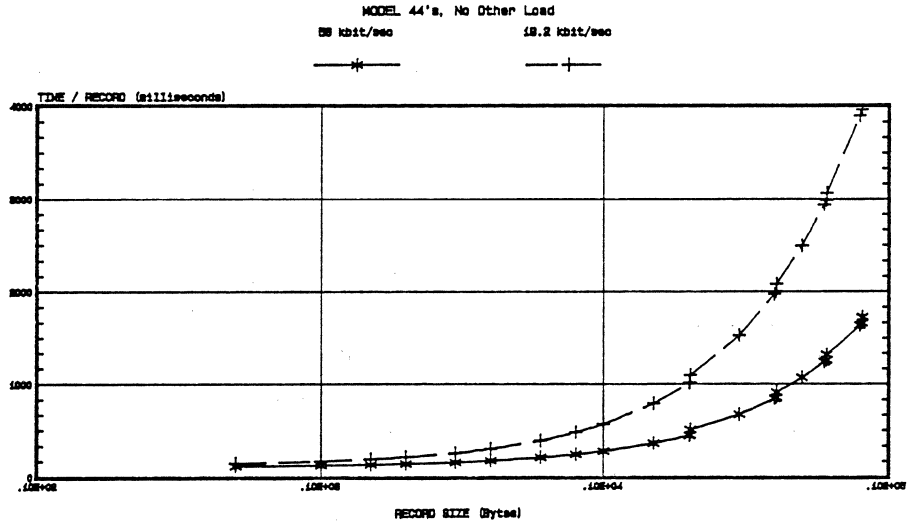


FIGURE 4. RFA PERFORMANCE

MODEL 44's, No Other Load

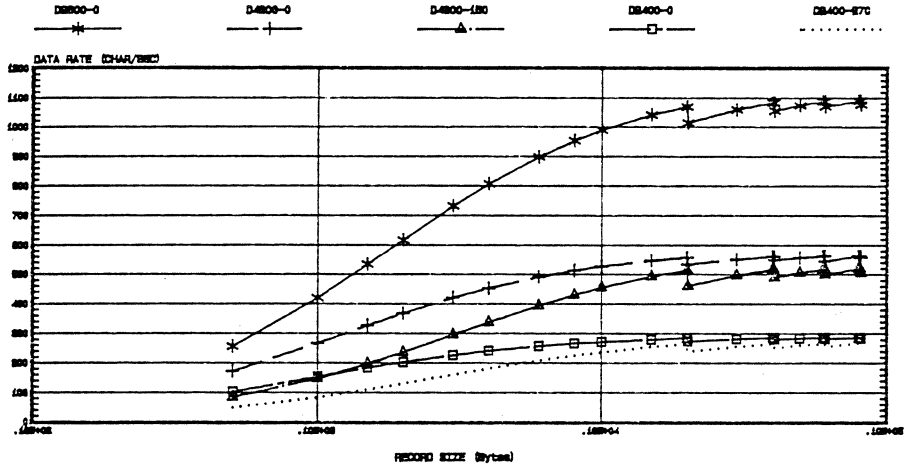


FIGURE 5. RFA TIME/RECORD

MODEL 44's, No Other Load

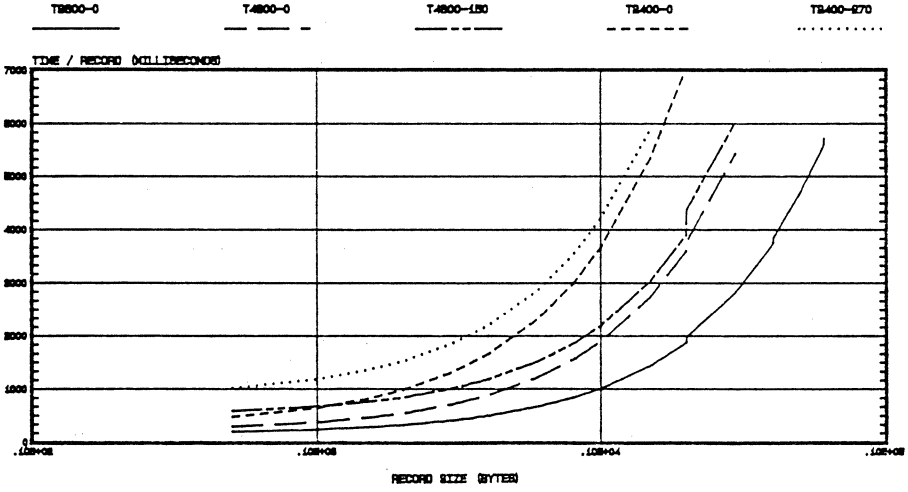


FIGURE 8. DSCOPY PERFORMANCE

MODEL 44's, no other CPU load

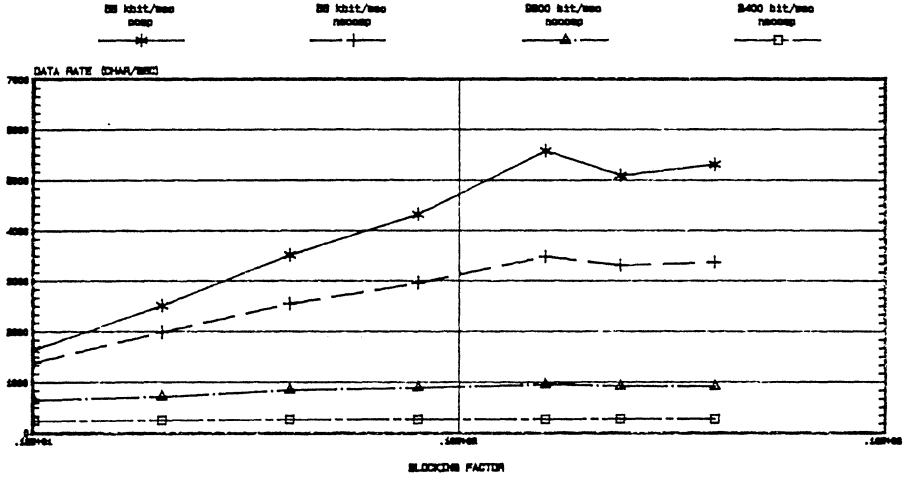


FIGURE 7, THROUGHPUT VS ERROR RATE

4800 bit/sec (150 msec) light CPU load

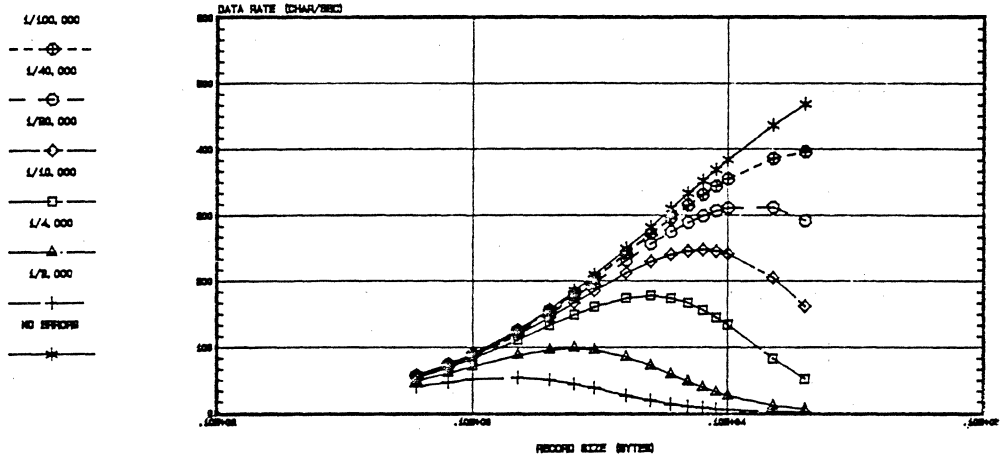


FIGURE 8, THROUGHPUT VS ERROR RATE

4800 bit/sec (150 msec) Heavy CPU Load

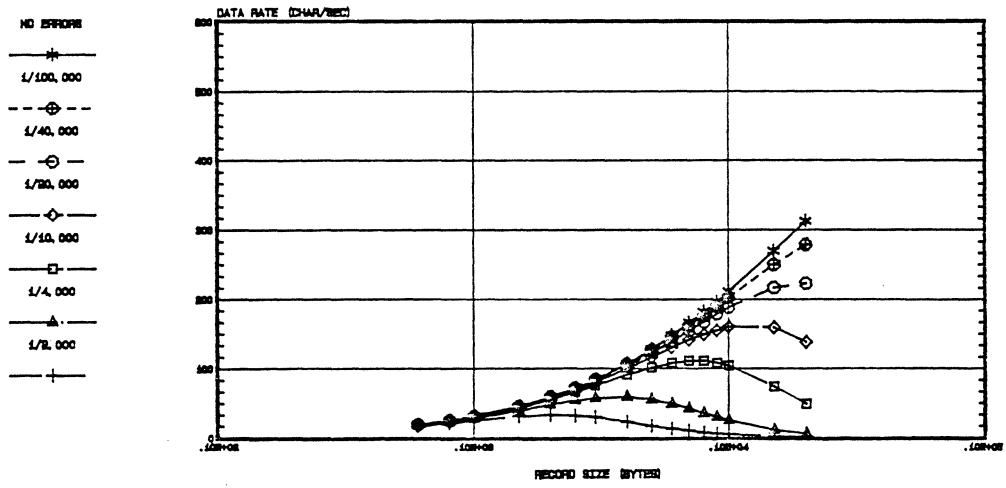


FIGURE 9. THROUGHPUT VS ERROR RATE

2400 bit/sec (150 msec) Light CPU Load

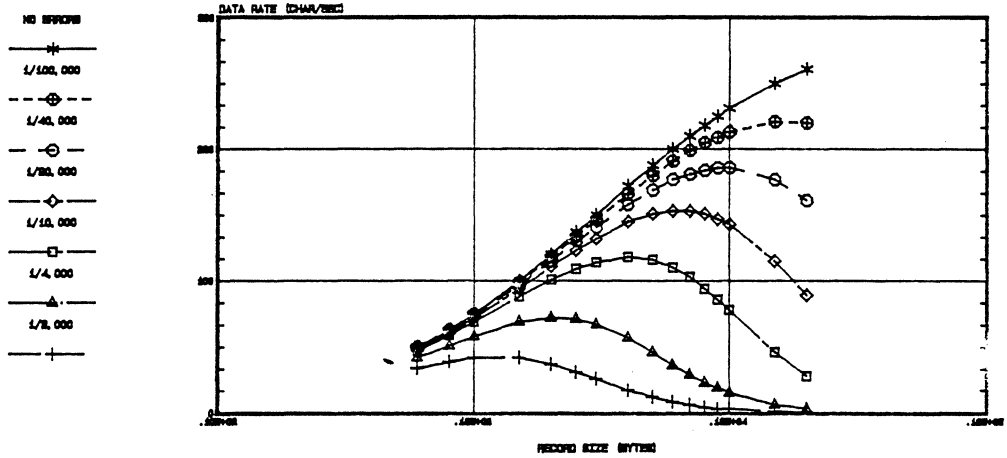
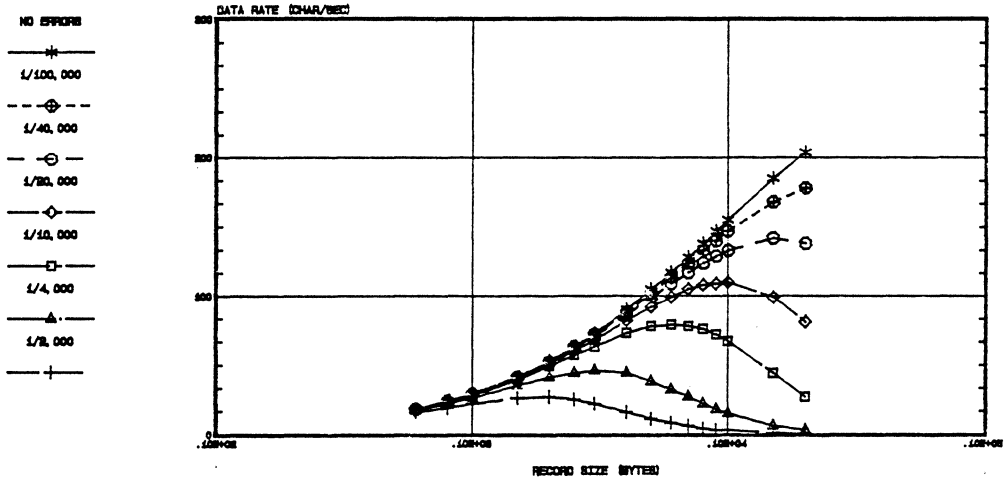


FIGURE 10. THROUGHPUT VS ERROR RATE

2400 bit/sec (150 msec) Heavy CPU Load



HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

INTRODUCTION TO LOCAL AREA NETWORKS

Marc Burch
Hewlett-Packard Company
Business Development Group
19447 Pruneridge Avenue
Cupertino, CA 95014

ABSTRACT

Local Area Networks (LAN's) are becoming extremely important in almost all areas of the business world today. The workplace is increasingly being filled with a wide range of computers, word processors, facsimile machines, terminals, copiers, printers, telephones, plotters, and personal workstations. We are fast approaching the maximum inflection point of being able to use all of this standalone and huge variety of equipment in an efficient and cost productive way. In order for groups of people in different organizations to function as efficient and well integrated units, the computers, peripherals and other equipment that they depend on must be able to communicate and exchange information quickly, easily, and reliably. An integrated total system approach based on local area networks offers a way to provide this communication exchange.

This paper provides a general introduction to local area networking basics, terminals, concepts, marketplace and installation considerations along with reviewing issues and trends in local networking.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

The potential market for Local Area Networks is very diverse and exciting. The market will be in four main areas - office environments, manufacturing plants and complexes, transaction services (Financial Distribution and Reservation systems) and educational establishments.

The largest market will be in the office environments of major corporations, where automation of office functions and increasing use of electronic storage media will require high capacity network systems with multiple channel capabilities. Every office in every company, from the smallest to the largest, in every industrial nation in the world is a potential candidate for a local area network. Most observers believe local networks will become as widespread in the office as the telephone system. Applications for intra-company networks include:

- *Data and File Exchanges between Workstations*
- *Peripheral Sharing*
- *Electronic Mail*
- *Access to Data Bases and Computing Power of a mainframe installation*
- *Video Conferencing*
- *Voice Store and Forward*
- *Others*

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

In manufacturing environments, local area networks will be increasingly used to connect terminals, office data and word processing equipment and process control and surveillance systems. These environments include:

- *Factory Data Collection*
- *Building/Environmental control and monitoring (Commercial Buildings, hospitals, schools, etc.)*
- *Process control, computer aided manufacturing (CAM) and Computer Aided--Design (CAD)*
- *Energy Managementations*
- *Aircraft Communications*
- *Photo composition*
- *Others*

Transaction services has a great need for local area networks to help with the following applications:

- *Financial Transactions*
- *Point-of-Sale Systems*
- *Reservation Systems (Hotels, Airline, etc.)*
- *Other*

In educational establishments, local area networks will be used to connect the ever increasing number of microcomputers in use for computer education, computer aided instruction (CAI) and research purposes. However, since most high schools and elementary school budgets are small, they typically purchase only two or three disks and a few printers, and make them available to all other computers via some type of local area network or multiplexing arrangement. This will change with the development of new storage and information sources that will significantly increase the use of computer/communication networks within the school system. Other applications inlcude:

- *Scientific*
- *Laboratory*

Several different technological developments allowed the evolution of new and exciting approaches to local area networking.

First, the much heralded advances in LSI (Large Scale Integration) and VLSI (Very Large Scale Integration) technology (Computers or Microprocessor chips) made it economically possible to distribute minicomputers and a whole array of intelligent, task-oriented peripheral equipment.

Second, many important gains in communication protocols have been combined with this very sophisitcated LSI and VLSI technology in nodes and network interfaces to provide the functions and performance levels needed for local area network communications.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

Third, much more of the lower level protocols responsible for interfacing to the network and controlling network functions are being designed into the network hardware. As a direct result of this:

- *Nodes have less overhead associated with network control and now can better perform their designated functions.*
- *The network hardware components can be mass produced in volume and at lowered cost, simplifying connection to the network and encouraging greater participation by users and equipment.*
- *Communications protocols that become standardized can be incorporated into the network hardware, thus allowing a variety of nodes from many different manufacturers to communicate without the need for expensive custom interfaces, and giving users increased vendor hardware and software independence.*

The trend today is towards greater numbers of separately identifiable computer based systems, due to the decrease in prices and computer processing components within an organization. This has caused a greater awareness (at both individual and organizational level) of the benefits of convenient interconnection of systems, often supplied by different manufacturers, to achieve coordinated access both to common resources (such as databases, analysis programs, development tools and office-style memos and reports) and to sophisticated or specialized (and therefore, expense) resources such as mainframe processor, file archive and management facilities, printers, plotters, etc. There is also a need for overall management control of system proliferation, duplication and/or dilution of effort, synchronization of activity and any other factors that can hurt an organizations resources.

Let's now look at what exactly is a local area network? Basically, a local area network is a data communications system that allows computers and peripherals to talk to each other over a common transmission medium.

What are local area network characteristics?

- *Reasonably high data transfer rates - one megabit/sec or higher. Usually, near 10 megabits/sec. Short transmission times make network operation transparent to users, allowing fast, multiple-station access.*
- *Limited geographical coverage - They usually have a diameter of a few kilometers, thus allowing rapid access and communication among distant organization groups.*
- *Low transmission error rates - Networks reliably accommodate heavy data transmission traffic should an error occur, a network station can detect it and institute a recovery.*
- *Low-cost connection and installation - You can attach or delete stations in the network without operational charges. Connection cost shouldn't exceed 10 to 20% of station-equipment costs.*

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

- *Large number of users - Networks can support tens of thousands of users and don't constrain organizational growth.*
- *Reliability and Availability - Networks can remain unaffected by individual failures or removals for service.*
- *Security - You can restrict network access to confidential, classified or sensitive data file.*
- *Flexible Topology - You can modify a network as the organization expands.*
- *Multimedia Communications - Some networks handle voice and video communications as well as data.*
- *Multivendor Compatibility - Networks can contain equipment from different vendors which allows greater functionality.*
- *Single Organization Ownership - Networks are usually owned by one organization and are designed by the organization to satisfy its needs. Gateways are used to communicate with other organizations.*

The design elements of a local area network must be carefully analyzed in order to provide the required levels of data communication capabilities and network performance. Anticipated use usually will determine which network type best matches your application requirement along with the following questions:

- *How access to the network and message traffic will be controlled*
- *How many and what kinds of nodes can participate and where they can be located*
- *How nodes will interact in the local area network - with other nodes and other networks*
- *The performance dynamics of the network - speed of response, ability to handle traffic loads*
- *The hardware and software that will be needed to implement the network, and all associated costs thereof*
- *The network applications that will be possible.*

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

However, before you can do your network and application planning, you need to understand the following:

1) Local Area Netowrk Topology
Point-to-Point or Multipoint

A. Ring
B. Star
C. Bus

2) Channel Access

A. Polling
B. Contention

3) Local Area Network Transmission Media

A. Twisted-Pair Wire
B. Coaxial Cable
C. Fiber Optics
D. Unbounded Medium (Radio, Microwave, and Infrared

4) Signalling Techniques

A. Baseband
B. Broadband

Local Area Network Topology

- *There are two kinds of links that serve as the building blocks of network topologies. Point-to-point and multipoint or multidrop. A Point-to-point links is a circuit which connects two (and only two) nodes without passing through an intermediate node. A multipoint or multidrop links is a single line which is shared by more than two nodes. Multipoint lines can be used to reduce the number of lines required to connect nodes and to reduce line costs.*

A. RING TOPOLOGY

The distinguishing feature of ring topologies is that nodes, which are connected by point-to-point links, are arranged to form an unbroken circular configuration. Transmitted messages travel from node-to-node around the ring. Each node must be able to recognize its own address in order to accept messages.

In addition, each mode serves as an active repeater, retransmission messages adddressed to other nodes.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

A. Ring Topology (Continued)

The need to retransmit each message can make ring nodes more complex than the passive nodes on a bus network. When Ring configurations are used to distribute control in local networks, access and allocation methods must be used to avoid conflicting demands for the shared channel. One way this is done by circulating a bit pattern, called a token, around the ring.

A node gains exclusive access to the channel when it grabs the token. It passes the right to access the channel (i.e. the token) onto other nodes when it is finished transmitting. Rings provide a common network channel wherein all nodes are fully connected logically. When control is distributed, each node can communicate directly with all other nodes under its own initiative.

Ring networks with centralized control are often called Loops. One of the nodes attached to the network controls access to and communication over the channel by the other nodes. Once a node is permitted by the control node to transmit a message, it can travel around the ring to its destination without further intervention by the control node.

In configuring ring networks, rings must be physically arranged so that they are fully connected. Lines have to be placed between any new node and its two adjacent nodes each time an addition is made. Thus, it is often difficult to prewire a building for ring networks in anticipation of nodes to be added in the future.

Failure of a node or an active component, adding a new node, or any other break in the ring confugration will almost always cause the network to stop functioning. Steps can be taken to allow bypass of failure points in distributed rings, although this usually increases the complexity of the repeater at each node, as well as the component costs. Failure of the control node in a centrally controlled ring would inevitably lead to network failure as well.

B. STAR TOPOLOGY

The distinguishing feature of Star is that all nodes are joined at a single point. Star configurations are frequently used for networks in which control of the network is located in the central node or switch. Point-to-point lines connect the central and outlying nodes, thus eliminating the need for the complex link and control requirements of other topologies.

A Star network could be constructed so that the control of communications would be exercised by one outlying node, or distributed generally to all outlying node, or distributed generally to all outlying nodes. In either case, the control function of the central node would be minimized. The node would serve as a simple switch to establish circuits between outlying nodes.

In all Star networks the central node is a single point of network failures. If it goes down, so does the entire network thus, reliability and the need for redundancy measures are important issues.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

B. Star Topology (Continued)

The Star is often used in timesharing applications, in PBX (Private Branch Exchange) telephone networks and in small clustered networks like word processing clusters.

C. BUS TOPOLOGY

The Bus topology functions in the same ways as a multipoint line and bus nodes share a single physical channel via cable tape or connectors. Messages placed on the bus are broadcast out and nodes must be able to recognize their own address in order to receive transmissions, however, unlike nodes in a ring, they do not have to repeat and forward messages. Therefore, there is none of the delay and overhead associated with re-transmitting messages at each intervening node, and nodes are also relieved of network control responsibility at this level.

Bus networks are easily configured and expanded and the network operation will continue in the event of node failures. Distributed Bus networks have been around for some time. This is attributed to the fact that the transmission media (usually coaxial cable or twisted pair wire) and transmission have been in use for some time by the cable and TV industry and for telephone and data communciations.

There are many considerations for the design of Bus networks. Components, such as transmitter/receivers, must be designed for reliable and mainframe operation. There must also be test equipment that will allow for fast and accurate Bus Fault detection and isolation to facilitate repair and maintenance.

CHANNEL ACCESS

A. POLLING

Polling Techniques determine the order in which nodes can take turns accessing the network, specifically so that direct conflict (i.e. collisions) between nodes is avoided. Polling is thus referenced to as an non contention method of network access.

Centralized polling may be based on a polling list with an arbitrary order (A,C,B,E,D..) or the order could reflect network node and traffic priorities. The order of access could also be based simply on the physical location of nodes.

Distributed Polling also allows control of access to the network by either token passing, slotted ring, or Cambridge ring. Token passing is a mechanism whereby each device, in turn and in a predetermined order, receives and passes the right to use the channel. Slotted rings emply a number of slots or frames of fixed size circulate around the ring. If a node chooses to transmit, it waits for a free or unused slot, inserts data into the appropriate field, and indicates the source and destination address. As in token passing nodes, along the way check to see if the frame is addressed to them. Due to the high speed of slotted rings certain inefficencies exist, such as only several bytes of data can be

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

A. Polling (Continued)

placed in each frame. Therefore, frames contain a high amount of control information vs data.

The Cambridge ring system is based on the establishment and use of circulating slots or packets of fixed size which are successively accessed, filled and read by network nodes as they pass by. Each packet slot travels in one direction only and make a complete revolution of the ring.

B. CONTENTION

Carrier sense multiple access with collision detect (CSMA/CD) anticipates conflicts or collisions and is designed to handle them. The multiple access feature of CSMA/CD allows any node to send a message immediately upon sensing that the channel is free of traffic. Therefore, you do not have the substantial portion of the waiting that is characteristic of non-contention techniques. This access control methods chief advantages lie in its simplicity reflected in lower cost per node because the scheme needs no complex priority access circuits - and its variable length message handling efficiently.

Carrier sense is the ability of each node to detect any traffic on the channel. Nodes will not transfer if they sense that there is traffic on the channel. Nodes will not transfer if they sense that there is traffic on the channel. However, collision could occur betwewen two messages due to the propagation delay (time it takes for the signal to travel across the network). As both nodes thought they had an open channel.

After detecting a collision, each node involved backs off, waits, and transmits again. This waiting period is usually random as it has proven to be more effective in auditing further collisions.

The most efficient use of CSMA/CD is when the packets are larger and therefore you have fewer collisions.

Several Bus networks used collision avoidance instead of CSMA/CD due to their lightly loaded applications. Collisions get detected by the nodes sending and receiving circuits. The sending node waits for an acknowledgement signal. If it does not come, it will retransmit until successful.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

3) LOCAL AREA NETWORK TRANSMISSION MEDIA

A. Twisted-Pair-Wire - Was one of the first wire types used in telephone communications and that is still true today. Pairs of wires are twisted together to minimize the interference created when adjacent pairs of wire are combined in multipair cables. Wire is usually made of copper and is inexpensive and easy to install. Used mainly in low speed data equipment with the maximum in the range of 9.6 Kbits per second. However the wire does emit and absorb high amounts of electrical interference, which can cause error rates.

Twisted pair provides a good media for integration voice and data through Digital Branch Exchanges.

B. Coaxial Cable - offers large bandwidth and the ability to support high data rates with high immunity to electrical interference and a low incidence of errors. Because it maintains low level capacitance in lengths to several miles, coax allows high megabit per second data rates without signal refeneration, echoes or distoration. Coaxial cable used for CATV (Community Antenna TV) has bandwidth in the range of 300-400 MHz.

C. Fiber Optics - Although expensive, possesses inherent local network point-to-point performance capabilities that outclass all other transmision media. Currently available fibers have usable bandwidth of up to 3.3 billion Hz, data rates of over one G-bits per second, high voltage isolation, non-electronic radiation, small size and light weight, and error rates are very low (one bit error per 109 bits).

However, fiber optics are still too costly and are also very hard to tape into to add additional nodes.

D. Unbounded Medium - Radio, Microwave and Infrared

Today, there are a few commercially available local area networks based on these unbounded medium technologies. Given time and the advancement of technology these three major types of signals will undoubtedly become more prevalent.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

4) SIGNALLING TECHNIQUES

a) Baseband

o Lower cost of cable used for interconnection
o Baseband interfacing cheaper than broadband modems
o Reasonably high data transmission rate
o Acceptably low error rate

Baseband local area networks are predominantly used for data communications and therefore the signals and transmission technology implemented are digital (optimized for data) with maximum capacity of approximately 10 MBPS. Most often 3/8 inch coaxial cable is used with Bus Topology with contention control - usually CSMA/CD.

b) Broadband

o High total bandwidth allows transfer of data, voice, video
o Total bandwidth can be (statically) Subdivided into many independent channels for transparent use by attached devices.
o The technology is common to CATV (Community Antenna TV), affording some cost-benefits with conventional CATV services.
o Acceptably low error rate for most applications
o No significant geographical extent limitation (up to 500 Km)

The advantages of broadband are to be found in applications which call for many point to point connections whre the interconnection pattern is essentially static or where very high point-to-point transfer rates (such as video) are required.

A broadband network uses analog transmission techniques and can accommodate up to 500 Mbps of information. Broadband networks use frequency diision multiplexing (FDM) to divide a single physical channel made of 1/2 inch 75 OHM CATV coaxial cable into a number of smaller independent frequency channels. These smaller channels can be allocated different bandwidths so that they can be used to transfer different forms of information - specifically voice, data and video.

Broadband shortcomings are that in addition to requiring network and station interfaces, it uses expensive fixed - frequency or frequecy agile (Tunable) modems costing $500 to $1200. In addition to easily doubling broadbands interface cost, tunable RF modems prove difficult to check, maintain and adjust because they are usually installed behind walls and above hung ceilings.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

b) Continued

Broadband networks must also rely on a central transmission facility or head end in a single cable network. This facility acts as the networks technical control center and filters incoming RF signals. But it also represents a point that could deactivate the entire network, if it fails.

The lack of industry wide accepted standards for interfacing equipment to broadband network is still a disadvantage. However, for Baseband this is not true as several vendors have endorsed the IEEE802 standard for Baseband networks.

In the final analysis, the choice between Baseband and Broadband is, and will continue to be, dictated by engineering economies.

FUTURE ISSUES FOR LOCAL AREA NETWORKS

Shared Resources - The proliferation of cheap intelligence is encouraging a drive towards shared resources rather than to shared logic. The number of individual devices that need to communicate will grow almost explosively over the next few years.

Wire-Less Connection and Satellites - Terminal equipment will be able to communicate with the local area networks by means of Radio links and infra red tranmissions.

Standardization - Starting to see some of this now (i.e. IEEE802 standard for Baseband) but the industry needs higher level standards to be set. The timescale for a satisfactory outcome of the various standardization activities is most likely to be on the order of 3-5 years.

Hardware Trends - Downward for prices as the combination of economics being achieved in packaging of electronic logic (LSI, VLSI, etc.) Along with new developments in information transmission media (Fiber Optics, Infrared) will ensure local area networks strong future.

Network Mangement - The growing dependence of large numbers of systems to local area networks will demand guarantees on networks availability and performance. This will have to be provided through effective resource management and planning.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

SUMMARY

The future for Local Area Networks looks exciting and wide open. Over the next two or three years considerable progress will be made in the development of local area networks. This is necessary if we are ever going to have the much heralded "office of the future". These developments would not just be to benefit the office but also where information is being handled - process industry, manufacturing, teaching, research, etc. However, the real challenge lies in being able to develop new applications that will take advantage of Local Area Networks.

REFERENCES

1. *Hopkins, G. and Meisner, N. "Choosing Between Broadband and Baseband Local Networks," Mini Micro Systems, June 82 pp 265-274*

2. *Kinnucan, P., "Local Networks Battle for Billion Dollar Market", High Technology, November/December 1981, pp 64-72*

3. *Heard, K. Local Area Networks, Gartner Group Special Report, Feb 1982*

4. *Digital, Introduction to Local Area Networks, 1982.*

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

6. *Strategic Incorporated, Intra-Company Networks: Broadband vs. Baseband, the Key Issues, February 1982.*

MPE DISC CACHE : IN PERSPECTIVE

John R. Busch
Alan J. Kondoff

Members of the Technical Staff
Hewlett Packard Corporation
Computer Systems Division
19447 Pruneridge Avenue
Cupertino, California
95014

ABSTRACT

MPE disc caching is a major new performance product under development for the HP 3000 family of computers. MPE disc caching effectively utilizes the excess main memory and processor capacity of the high-end 3000 family members to eliminate a large portion of the disc access delays encountered in an uncached system. With disc caching, disc data is available at main memory rather than disc access delays with a probability that increases with main memory size. The MPE disc cache designers present an overview of the purpose of disc caching, its design approach, its advantages over the alternatives, and its impact on the system price/performance of the HP 3000 family. The full text of this paper will be distributed at the conference.

PLANNED PERFORMANCE AND CAPACITY ENHANCEMENTS FOR HP 3000 SYSTEMS

Jeff Byrne
Bill Walker
Hewlett-Packard Co.
Computer Systems Division

The tremendous success of the HP 3000 stems from the strategy upon which it was founded and which continues to guide the HP 3000 development today. For HP 3000 processors and MPE, this strategy translates into developing an increasingly broad range of processor price and performance while maintaining MPE compatibility. Combined with HP's upgrade program, this broad compatible family protects our customer's investment in HP 3000 hardware and software.

Implementation of this strategy is reflected in the evolution of the HP 3000 family. Approximately every two years, we have introduced a new model doubling the performance of our most powerful HP 3000 system. Increasing performance has also been accompanied by a large expansion in the configuration capacity of our systems. Throughout all of these changes, we have maintained upward MPE compatibility.

This continual broadening of the range of information management applications addressed by the HP 3000 has created new needs for performance and capacity in our systems. As processors become faster, the speed of I/O access can become a limiting factor on overall system performance. In addition, expanding configurations and a broader range of applications have caused some customers to encounter limitations imposed by the size of tables in MPE. In response to these needs, we will be adding new products and enhancements to HP 3000 systems in the near future. These planned products and enhancements will be described in this presentation.

Advanced Techniques Using VPLUS

by
Michael A. Casteel
Vice President
Computing Capabilities Corporation
Mountain View, California

Introduction

This paper is intended to cover two particular topics of interest to a number of VPLUS users. It will cover procedures for alternating between VPLUS block mode, using formatted screens, and conversational mode such as used with MPE and utilities. This technique may be used to integrate existing conversation mode dialogues with new VPLUS applications. Perhaps more generally, it can be of tremendous value when debugging VPLUS applications, especially when two terminals are not available within arm's reach.

Also presented are procedures for printing the screen contents, either to an attached or integral printer or to the system printer. Of course, these procedures are not specific to VPLUS applications and as such may be of even broader utility.

Both of the particular topics of this paper involve communication with and control of the terminal. It is important, therefore, to cover some background material first, in order to understand the terminal configuration and communications protocol in effect in the VPLUS environment. Printing the screen or switching from block/format to conversational mode are not particularly difficult, once the VPLUS operating mode is understood. Happily, all VPLUS-supported CRT terminals are basically compatible. There are some significant differences between point-to-point and multipoint (MTS) operation, which will be covered in the discussion.

Terminal and I/O Configuration

The information in this section has been deduced from assorted terminal and software manuals, observations and conversations. It can't all be guaranteed correct, but has so far proven out in those cases where it was needed.

Basic Terminal Configuration

A few of the Keyboard Interface straps (options) must be set in a particular way for VPLUS operation. For 2640B or 2644 terminals you must do this manually, by opening the terminal and setting switches. VPLUS sets the others automatically during VOPENTERM or VGETNEXTFORM with $REFRESH. Special point-to-point options are:

D - open (Line/Page) Establishes Page mode, whereby the ENTER key transmits the entire screen instead of only a single field or line.

E - open (2640B) Allows the terminal function keys (f1-f8) to be used without holding the CNTL key. (Optional)

F - open (2640B) Provides for 2645-compatible handshake protocol (DC1/DC2/DC1).

G - open (InhHndShk) Provides that computer-requested block transfers (e.g. terminal status, cursor sense, printer command response) observe the DC1/DC2/DC1 protocol.

The main strap of interest in multipoint (MTS) is:

J - closed (Auto Term) MTS opens this strap, which has the effect of limiting data transmission to data on the screen above the cursor position at the time the ENTER key is pressed. VPLUS closes this strap to allow transmission of all data in the form.

There are other straps which are important to VPLUS operation, but the "normal" setting is the appropriate one.

In addition, VPLUS requires that the terminal be set for block mode. This is the normal mode in MTS In point-to-point, VPLUS will set the terminal in block mode automatically (except 2640/44).

Terminal File Configuration

VPLUS controls and communicates with the terminal through the MPE file system using the standard set of intrinsics. Some special file system parameters are set for the point-to-point and multipoint device drivers under VPLUS, generally via the FCONTROL intrinsic. Significant FCONTROL

functions for point-to-point operation are:

13 - Echo is turned off (if it isn't already)
25 - Set alternate terminator to RS
31 - Enable VPLUS driver control
38 - Set terminal type to 10 (if it isn't already).

Character Echo (FCONTROL 12/13)

Normal full-duplex point-to-point operation requires the HP3000 to echo each character it receives back to the terminal to be displayed. In block mode, under VPLUS, each character is displayed on the screen as you type it, and nothing is sent to the computer until you press ENTER. Since everything you type is already shown on the screen, a computer echo of the block transmission would only confuse matters. Therefore, FCONTROL 13 (disable echo) is used.

Alternate Terminator (FCONTROL 25)

VPLUS uses the "ultimate" form of block mode on HP terminals, i.e. Block/Page mode. This allows the terminal to send the whole screen at once, the most efficient form of block transmission. However, full page, block mode inputs do not end in the usual carriage return (CR) code which terminates other inputs; in fact, there may be a number of carriage returns in the midst of the input. Instead, HP terminals send a control code called "Record Separator" (RS) to signal the end of the block. VPLUS accommodates this by setting the RS code as the "alternate terminator" in point-to-point operation, using FCONTROL 25.

Terminal Type (FCONTROL 38)

Terminal ports used by HP CRT terminals are usually configured as Terminal Type 10, which signals the I/O driver to observe certain protocols which are appropriate to such terminals. For example, the driver will send a control code, ENQ, after every 80 characters output. HP terminals will answer with an ACK when they are ready for the next 80 characters. This is the famous ENQ/ACK handshake. When you use VPLUS you must be using an HP compatible terminal; so VPLUS sets the Terminal Type to 10 using FCONTROL 38, in order to activate these protocols.

VPLUS Driver Mode (FCONTROL 30/31)

FCONTROL 31 is more puzzling, since HP has so far neglected to document it. When reading from the terminal under this option, a DC2 code as the first character received causes the MPE device driver to set up a block read. This is necessary since, when you press ENTER, the terminal doesn't just send the screen contents as a big block of data. It only sends the single control code, DC2, and waits for a DC1. VPLUS used to handle the DC2 itself, but now uses the new Driver Mode. Now, on receipt of the DC2, the MPE device

driver (not the program, not VPLUS) responds with:

<esc>c<esc>H<DC1>

where <esc> stands for the ASCII 'ESCAPE' control code, and <DC1> for the DC1 code. These control codes lock the keyboard, home the cursor, and trigger the block data transfer. The driver times the block read, in case the terminator character is lost. The read terminates by character count, timeout or receipt of an RS code (the VPLUS alternate terminator, i.e. the Block/Page Mode terminator character). It is natural to assume that the block read functions armed with FCONTROL 31 have been moved into firmware on the ATP.

MTS Unedited Input (FCONTROL 41)

The above functions are applicable to point-to-point terminals. For multipoint terminals, the MTS driver can handle block transmissions pretty much as usual. The only important difference to the driver is one FCONTROL:

41 - Set unedited mode (with parameter %137)

This option is used to stop MTS from placing a block delimiter everywhere VPLUS puts the cursor. Block delimiters set by MTS (or the ENTER key with the terminal's strap J open) would prevent the transmission of the full screen to the computer.

User Read Time-out (FCONTROL 4)

For point-to-point or multipoint, VPLUS will also use FCONTROL:

4 - Enable read time-out

when OPTIONS (word 56 of the Comarea) has bits 9-10 set to 01. USER'TIME (word 58) gives the number of seconds to allow for input.

Conversation vs. VPLUS Mode

There are occasions in many applications where it is desirable to remove the terminal from VPLUS operation for a while, then resume VPLUS. For example, you may wish to run another program which doesn't use VPLUS, or just engage in a conversational dialogue. Debugging, with DISPLAYs and ACCEPTs or PRINTs and READs, or using MPE Debug or Toolset, is an almost universal occasion for conversation mode.

Normally, a program resumes conversation mode with a call to VCLOSETERM when it terminates. This suggests an approach to switching modes: To change from VPLUS mode to conversation mode, call VCLOSETERM. To switch back, call VOPENTERM. Since VCLOSEFORMF isn't called, a lot of valuable information is preserved:

Current form name
Data buffer contents
Next form name
Repeat/Next form options
Screen label settings
Save field contents

This may be the best approach to use when you wish to switch modes in order to run another program. There's no telling in what state the other program will leave the terminal and I/O configuration, but VOPENTERM should be able to sort it out. In fact, the undocumented intrinsics VTURNOFF and VTURNON should have about the same effect, without taking quite such drastic steps as closing and re-opening the terminal file, clearing the screen, and so on. Parameters are the same as VCLOSETERM and VOPENTERM, respectively.

If you take this approach, you will discover a few complications which may (or may not) affect your application:

1. Your screen remains empty.

   VOPENTERM clears the display, but VSHOWFORM doesn't know it. This means that your next call to VSHOWFORM may not write any data to the screen, since VPLUS believes the last form to be still there. This is a result of VSHOWFORM optimization, which you can correct by moving 7 to word 34 of the Communications Area (SHOWCONTROL) before calling VSHOWFORM. Don't forget to reset SHOWCONTROL afterward (New requirement with the Q MIT). Or, you may call VGETNEXTFORM with Next Form name $REFRESH.

2. Your screen is still empty.

   VOPENTERM reconfigures workspaces on a 2626 using local form storage, but VSHOWFORM doesn't know it. In this

case, moving the 7 to SHOWCONTROL might just cause VSHOWFORM to try to redisplay the form from the workspace where it used to be. You need to use $REFRESH in this case, or perhaps VLOADFORMS with SHOWCONTROL. As of this writing, it is too early to tell the effect on 2624B Local Forms Storage.

3. Your screen labels are missing.

   VCLOSETERM removes your screen labels from the terminal, but VOPENTERM doesn't put them back. It is anticipated that you can use $REFRESH to correct this problem in the Q MIT. We can only hope that VOPENTERM will also be corrected. To solve this problem prior to the Q MIT, follow VOPENTERM with VSETKEYLABELS, for global or form labels, whichever is appropriate. I use VGETKEYLABELS to retrieve the needed values, so they don't need to be coded into the program.

   You may actually have to do two VSETKEYLABELS, due to VPLUS optimization. If you simply set them to the same value they had, VSHOWFORM will not bother to put them on the screen. I get around this by first setting the labels to a different value, then setting them back.

   Note: As of VPLUS B.02.02, you must do your VGETKEYLABELS before calling VCLOSETERM, since the latter destroys the values in the VPLUS label buffer.

Switching Modes Yourself

Most often all you want to do is switch modes quickly in order to display a message or allow some form of debugging dialogue. For example, the VPLUS debugging facilities in INSIGHT II and RADAR (two Computing Capabilities Corporation products) keep the terminal in conversation mode except during VPLUS output or input. As a result, Debug breakpoints, abort messages and VPLUS screens are all accommodated on a single terminal.

This is not a difficult task, as it turns out, made easier since the addition of FCONTROL 30/31 (VPLUS Driver Mode) for point-to-point terminals. Appendix 1 includes a listing of a short SPL procedure, VSETMODE, which performs the necessary functions. This procedure has two parameters: The VPLUS Communications Area (just like VPLUS intrinsics) and an integer mode. Mode zero calls for VPLUS operation, and one calls for conversational mode.

This routine operates outside the bounds of documented VPLUS operations. This means it is possible that HP could change things around in such a way that it won't work. This is not highly likely, since the operations performed are so fundamental.

VSETMODE Comarea Usage

You will see that VSETMODE uses three words in the VPLUS Comarea. The first two are the terminal file number and the terminal model as determined by VPLUS. Both are documented in the current edition of the VPLUS Reference Manual.

The third word used is not documented in the VPLUS manual. Bits 4-9 of this word contain the original MPE Terminal Type. We need this to learn if the terminal is an MTS terminal, designated by Terminal Type 14. This (MTS) information could also be obtained using FCONTROL 39 on the terminal file, without reference to this word.

Bit 1 is used to determine whether character echo was on before VOPENTERM. Half-duplex connections, for example, should not have echo turned on. The essential information could be obtained without reference to this word by using FGETINFO to check for a half-duplex subtype.

VSETMODE Operation

To enter conversational mode, VSETMODE first conditions the terminal by writing a sequence of control commands. This begins by turning off Format Mode, moving the cursor to the bottom of memory, and unlocking the keyboard:

<esc>X<esc>F<esc>b

On terminals which support this operation, VSETMODE also turns off block mode:

<esc>&k0B

On the new line of terminals, the aids and modes keys will be unlocked:

<esc>&jR

When returning to VPLUS mode, these operations are reversed, although the keyboard, aids and modes keys are not locked.

If character echo was on before VOPENTERM, it will be turned on again by VSETMODE when entering conversation mode and off when resuming VPLUS mode (FCONTROL 12 and 13, respectively).

Since the introduction of VPLUS Driver Mode for point-to-point terminals, it is no longer necessary to change other MPE file parameters. Your terminal will operate normally unless you both:

1) Read input using the terminal file number in the VPLUS Comarea.
2) Send a DC2 code to the computer (for example, press ENTER or a function key).

If you do both these things you may find that your terminal locks up. This is because the MPE device driver locks your keyboard when it receives the DC2. To free up the terminal you will have to:

1) Unlock the keyboard, e.g. soft reset.
2) Type an RS control code (control-^) to end the read.

If your terminal is connected via MTS rather than point-to-point, VSETMODE has to change another file parameter. In order to enter conversational mode, Unedited Mode (FCONTROL 41) set by VOPENTERM must be turned off. This is done by FCONTROL 41 with a parameter of zero. When resuming VPLUS mode, Unedited Mode must be turned on (parameter octal 137), and the terminal's J strap must be reset.

Printing the Screen Contents

There are many occasions when it would be useful to print the screen contents. There is a VPLUS intrinsic, VPRINTFORM, whose purpose is to print a copy of the current form and its contents. However, since this intrinsic only prints to a file, it is mainly useful for obtaining listings on a system printer.

If the screen to be printed consists of several forms, one call to VPRINTFORM will not print the entire screen. Instead, you must have the foresight to call VPRINTFORM as each form is displayed, thus piecing together a screen image in the print file at the same time it is being assembled on the terminal.

Printing to a Terminal Printer

A more direct approach is to perform a full screen print operation. The simplest way is to equip your terminals with local printers, either integrated or attached. It then becomes a matter of commanding the terminal to copy the screen to the printer. This produces a true hard copy of what was actually displayed. Also, the only foresight you need is to provide a routine which will issue the right command when needed. Appendix 2 includes a listing of an SPL procedure, VPRINTLOCAL, which will print the terminal screen to a terminal printer. This procedure requires only one parameter, the VPLUS Comarea.

My usual practice is to assign one of the terminal function keys as a PRINT key. Whenever the program accepts input through VREADFIELDS, I check word 6 (LASTKEY) in the VPLUS Communications Area. If the user pressed the PRINT key, call the Print routine, and when finished loop back to call VSHOWFORM and VREADFIELDS once more. VSHOWFORM is called to unlock the terminal keyboard which was left locked by VREADFIELDS. It also serves to display any message sent to the window by the Print routine.

VPRINTLOCAL Operation

VPRINTLOCAL uses two different approaches to performing its function. This is because there are two forms of print command recognized by HP terminals: a simple "dump the screen" available on 2640B and 262X terminals, and the more complicated device control sequence needed for the 2645-based terminals. Except on the latter, all it takes to "dump the screen" is to command the terminal to:

1) Turn off Format Mode
2) Copy the screen to the printer
3) Turn on Format Mode.

This is a matter of sending six characters to the terminal:

<esc>X<esc>0<esc>W

The actual 'screen print' command is the <esc>0 (that's the numeral zero). You need to turn Format Mode off in order to copy the protected areas of the form. If Format Mode is on when using an attached printer, the terminal assumes that the printer contains a preprinted form matching the screen. Only the data in unprotected fields will be printed.

VPRINTLOCAL sends this command using FWRITE to the terminal file number in the Comarea (word 49, FILEN). This takes advantage of the VPLUS Driver Mode in effect on this file, and will work even if the terminal is not the job/session logon device. This FWRITE includes carriage control code %320 (octal 320, decimal 208) as its fourth parameter. This value suppresses the carriage return and line feed which normally follow every output.

Printing on a 2645 is more complicated in two ways: first, the command sequence is longer; worse, the terminal insists on talking back. You have to program a dialogue with the terminal. VPRINTLOCAL starts by sending the operative command:

<esc>X<esc>H<esc>b<esc>&p3s4dM

This means,

<esc>X	Turn off Format Mode
<esc>H	Home the cursor
<esc>b	Unlock the keyboard. This allows the user to cancel the print operation by pressing the Return key.
<esc>&p3s4dM	Copy everything (M) from the display (3s) to the printer (4d) starting from the cursor position.

Note: This command will also work on the 262x terminal family. The '4d' code normally specifies the external (attached) printer. To use the integral printer, use '6d' or specify 'DeviceCode4' on the terminal's Configuration Menu as the INT printer.

This starts the printing process, if the terminal has an attached printer and the firmware to support it. Once the terminal has completed printing, or determined that it couldn't print, it will want to send a single character response:

S - Print operation completed.
U - Operation aborted by the user. (User pressed Return)
F - Print not completed (out of paper, etc.).

VPRINTLOCAL must read this response, or the next VREADFIELDS will read it as if the user sent it with the ENTER key. This is done in the subroutine RESPONSE, using a timed FREAD (FCONTROL 4 before FREAD) in case something goes wrong. 60 seconds are allowed, which should be enough to print a screen unless you're using a very slow printer.

> Note: Under VPLUS Driver Mode (FCONTROL 31), the terminal's response will cause the MPE device driver to lock the keyboard. You will need to call VSHOWFORM in order to unlock it before the next VREADFIELDS.

Finally, after receiving the terminal's response, turn format mode back on:

```
<esc>W
```

In order to determine which type of terminal it is dealing with, VPRINTLOCAL looks at VPLUS Communications Area word 59 (IDENTIFIER). Values of 1,8,9,11 or 13 signify 2640B or 262X terminals.

Screen Copy on 2626

Users with 2626 terminals may wish to use the 'Screen Copy' device control operation. This function is performed using the device control sequence:

```
<esc>&p4dE
```

Like the device control sequence on a 2645, this command produces a response which must be processed by the computer. The 2626 Screen Copy includes the screen labels on the printed output.

Application Notes

VPRINTLOCAL uses the more complicated device control sequence on everything but a 2640, which doesn't support it. This is because I want the printer to do a page eject after printing the screen, but this doesn't happen on my 2624 terminal if I just give it an <esc>0. The page eject results from the command <esc>&p4u5C.

My Direct 825 terminal pretends that it is a 2622, but does not recognize the device control sequence. Since VPRINTLOCAL sends the device control sequence to 2622s, the 825 gets confused. It's easy to command the terminal PRINT function from the keyboard on the 825, though: just Function/Print (9 on the numeric keypad). No need to take the terminal out of format mode, and it does a page eject, too.

Printing to a System Printer

It takes more work than printing to a terminal printer, but you can copy the screen to a file or system printer if you

wish. The well-known program PSCREEN performs such a function. Appendix 3 includes a listing of a procedure, VPRINTSCREEN, which has been tailored to perform this function in the VPLUS environment, for both point-to-point and MTS terminals. This procedure accepts two parameters: The VPLUS Comarea and an integer carriage control code, which will be written after the screen has been printed. The value 49, for example, will produce a form feed.

Like VPRINTFORM, VPRINTSCREEN will print to any file if you place its MPE file number in word 36 of the VPLUS Comarea (PRINTFILNUM). If this word is zero, VPRINTSCREEN will open a file named FORMLIST on device LP, print the screen and close the file.

VPRINTSCREEN Operation

There are two operations of special interest in VPRINTSCREEN: Reading the screen contents and stripping control codes from the data before printing. These operations are performed in the subroutines READ'SCREEN and PRINT'LINE, respectively.

Reading the Screen

Reading the screen is a simple matter since the addition of VPLUS Driver Mode for point-to-point terminals. In order to read the entire screen, the terminal must first be removed from format mode. All that is required is the command string:

<esc>X<esc>d

After format mode has been turned off, the <esc>d commands the terminal to transmit the contents of its memory to the computer. The data is then read using FREAD. Note that VPRINTSCREEN uses the terminal file number in the VPLUS Comarea. This takes advantage of the VPLUS Driver Mode in effect on this file, and will support terminals which are not the job/session logon device.

The last character read is the block terminator character, RS for point-to-point or GS for MTS. The subroutine replaces this by a Carriage Return followed by RS for consistency.

Stripping Control Codes

The characters received from the terminal include not only form and field data but display control codes, such as:

<so>	Shift to alternate character set
<esc>[	Start field
<esc>&d<letters>	Display enhancement

Each screen line is terminated by a Carriage Return.

The PRINT'LINE subroutine scans the characters in the line, skipping control codes, such as <so>, and escape sequences (starting with <esc>). There are two kinds of escape sequences: Standard (two characters), such as <esc>[, and generic, such as <esc>&d<letters>. If the character following <esc> signifies a generic escape sequence, PRINT'LINE skips to the terminator character, which is either an '@' or uppercase letter. During the scan, PRINT'LINE moves the printable characters to the start of the buffer, to ensure that the print line begins on a word boundary for FWRITE.

VPRINTSCREEN does not suppress printing of Security Video fields, which do not display on the screen. If you wish, you may add the logic yourself ¬(send me a copy). In general, characters displayed in an alternate character set (i.e. following <so>) should not be printed either.

Acknowledgements

Thanks to Ross Scroggs for first documenting the precise effects of FCONTROL 30/31, and to Mark Cousins of CCC for the original version of VPRINTSCREEN. VSETMODE and VPRINTLOCAL have been lifted, almost bodily, from INSIGHT II, a VPLUS transaction processor.

Should the reader discover any improvements or corrections to any of these routines, in any environment (DS/1000?), the author would appreciate hearing from you.

APPENDIX 1. VSETMODE Procedure.

This SPL procedure will switch the user's terminal between VPLUS block/format mode and MPE conversational mode.

```
PROCEDURE VSETMODE(COMAREA, MODE);
VALUE MODE;
INTEGER MODE;                    << 0=VPLUS ; 1=CONVERSATION >>
INTEGER ARRAY COMAREA;
BEGIN

ARRAY BUF(0:19);                        << LOCAL OUTPUT BUFFER >>
BYTE ARRAY BBUF(*)=BUF;
INTEGER LEN;

DEFINE CONV      = (MODE=1)#;
DEFINE COM'TERM = COMAREA(48)#,    << TERMINAL FILE NUMBER >>
       COM'IDENT= COMAREA(58)#,          << TERMINAL MODEL >>
       COM'TYPE = COMAREA(49).(4:6)#,     << MPE TERM TYPE >>
       COM'ECHO = COMAREA(49).(1:1)#;       << WAS ECHO ON >>

INTRINSIC FWRITE, FCONTROL;

<<  FIRST, CONDITION THE TERMINAL >>

IF COM'IDENT>2 THEN                    << PROGRAMMABLE STRAPS >>
   IF CONV THEN MOVE BBUF := (27,"X",27,"F",27,"b",
                              27,"&k0B"),2
   ELSE MOVE BBUF := (27,"W",27,"&k1B"),2
ELSE                                                 << 2640/44 >>
   IF CONV THEN MOVE BBUF := (27,"X",27,"F",10,
                              27,"bUNLATCH BLOCK MODE"),2
   ELSE MOVE BBUF := ("LATCH BLOCK MODE",27,"W"),2;
IF CONV AND COM'IDENT>=8 THEN MOVE * := (27,"&jR"),2;
LEN := TOS-LOGICAL(@BBUF);
FWRITE(COM'TERM, BUF, -LEN, %320);

<<  NEXT, RESET CHARACTER ECHO >>

IF COM'ECHO=0 THEN
   FCONTROL(COM'TERM, (IF CONV THEN 12 ELSE 13), LEN);

<<  FINALLY, TAKE CARE OF MTS >>

IF COM'TYPE=14 THEN BEGIN                                 << MTS >>
   LEN := IF CONV THEN 0 ELSE %137;
   FCONTROL(COM'TERM, 41, LEN);
   MOVE BBUF := (27,"&s0J"),2;
   LEN := TOS-LOGICAL(@BBUF);
   IF NOT CONV THEN FWRITE(COM'TERM, BUF, -LEN, %320);
   END;
END;
```

APPENDIX 2. VPRINTLOCAL Procedure.

This procedure prints the screen contents by commanding the CRT terminal to copy the screen to an attached or integral printer.

```
PROCEDURE VPRINTLOCAL(COMAREA);
INTEGER ARRAY COMAREA;
BEGIN

ARRAY BUF(0:15);                          << LOCAL I/O BUFFER >>
BYTE ARRAY BBUF(*)=BUF;
INTEGER TIME, LEN;

DEFINE COM'STATUS    = COMAREA#,              << STATUS WORD >>
       COM'FILERRNUM= COMAREA(36)#,     << MPE FILE ERROR >>
       COM'TERM      = COMAREA(48)#,    << MPE FILE NUMBER >>
       COM'IDENT     = COMAREA(58)#;     << TERMINAL MODEL >>

INTRINSIC FCONTROL, FWRITE, FREAD, FCHECK;

LOGICAL SUBROUTINE RESPONSE;  << DEVICE CONTROL RESPONSE >>
BEGIN
TIME := 60;
FCONTROL(COM'TERM, 4, TIME);
FREAD(COM'TERM, BUF, -2);
IF <> THEN
   BEGIN
   FCHECK(COM'TERM, COM'FILERRNUM);
   IF COM'FILERRNUM<>31 THEN COM'STATUS := 160;
   END;
IF COM'STATUS=0 THEN
   IF BBUF<>"S" AND BBUF<>"U" THEN COM'STATUS := 1234;
RESPONSE := COM'STATUS=0;
END;

IF COM'STATUS<>0 THEN RETURN;          << JUST LIKE VPLUS >>
IF COM'IDENT<3 THEN           << USE SIMPLE PRINT COMMAND >>
   BEGIN
   MOVE BBUF := (27,"X",27,"0",27,"W"),2;
   LEN := TOS-LOGICAL(@BBUF);
   FWRITE(COM'TERM, BUF, -LEN, %320);
   END
ELSE                          << USE DEVICE CONTROL COMMAND >>
   BEGIN
   MOVE BBUF := (27,'X",27,"H",27,"b",27,"&p3s4dM"),2;
   LEN := TOS-LOGICAL(@BBUF);
   FWRITE(COM'TERM, BUF, -LEN, %320);
   IF NOT RESPONSE THEN RETURN;
   MOVE BBUF := (27,"&p4u5C"),2;              << FORM FEED >>
   LEN := TOS-LOGICAL(@BBUF);
   FWRITE(COM'TERM, BUF, -LEN, %320);
   RESPONSE;
   END;
END;
```

APPENDIX 3. VPRINTSCREEN Procedure.

This procedure prints the screen contents to a file named FORMLIST on device LP. Escape sequences and control characters are stripped out of the data before printing.

```
PROCEDURE VPRINTSCREEN(COMAREA, PAGECTL);
INTEGER ARRAY COMAREA;
INTEGER PAGECTL;
BEGIN

ARRAY BUF(0:2047);                          << LOCAL I/O BUFFER >>
BYTE ARRAY BBUF(*)=BUF;
LOGICAL LOCAL'FILE := FALSE;
BYTE POINTER BUFCHAR,
             LINECHAR;
INTEGER LEN;

EQUATE LF   = 10,
       CR   = 13,
       RS   = 18,
       ESC = 27;

DEFINE COM'STATUS       = COMAREA#,
       COM'PRINTFILNUM = COMAREA(35)#,
       COM'FILERRNUM    = COMAREA(36)#,
       COM'TERM         = COMAREA(48)#;

INTRINSIC FOPEN, FREAD, FWRITE, FCHECK, FCLOSE;

SUBROUTINE READ'SCREEN;              << READ SCREEN CONTENTS >>
BEGIN
MOVE BBUF := (27,"X",27,"d"),2;
LEN := TOS-LOGICAL(@BBUF);
FWRITE(COM'TERM, BUF, -LEN, %320);
LEN := FREAD(COM'TERM, BUF, -4094);
IF <> THEN
   BEGIN
   FCHECK(COM'TERM, COM'FILERRNUM);
   IF COM'FILERRNUM<>31 THEN COM'STATUS := 160;
   END;
IF LEN>0 THEN LEN := LEN-1;              << STRIP TERMINATOR >>
BUF(2047) := [8/27,8/"W"];             << RESTORE FORMAT MODE >>
FWRITE(COM'TERM, BUF(2047), -2, %320);
MOVE BBUF(LEN) := (CR, RS);      << SET STANDARD TERMINATOR >>
END;
```

```
SUBROUTINE FILE'ERROR;
BEGIN
IF COM'STATUS<>0 THEN RETURN;
FCHECK(COM'PRINTFILNUM, COM'FILERRNUM);
COM'STATUS := IF COM'PRINTFILNUM=0 THEN 190 ELSE 191;
END;

SUBROUTINE START'FILE;              << OPEN COM'PRINTFILNUM >>
BEGIN
IF COM'PRINTFILNUM<>0 THEN RETURN;
MOVE BBUF := "FORMLIST LP ";
COM'PRINTFILNUM := FOPEN(BBUF, %507, %4, -251, BBUF(9));
IF COM'PRINTFILNUM=0 THEN FILE'ERROR
ELSE LOCAL'FILE := TRUE;
END;

SUBROUTINE PRINT'LINE;          << STRIP CONTROLS AND PRINT >>
BEGIN
@BUFCHAR := @BBUF;
WHILE LINECHAR<>CR DO
    IF LINECHAR=ESC THEN                    << ESCAPE SEQUENCE >>
       IF "&"<=INTEGER(LINECHAR(1))<="*" THEN
          DO @LINECHAR := @LINECHAR(1)
          UNTIL "@"<=INTEGER(LINECHAR(-1))<="Z" OR
                LINECHAR=CR
       ELSE @LINECHAR := @LINECHAR(2)
    ELSE
       IF LINECHAR<" " THEN @LINECHAR := @LINECHAR(1)
       ELSE
          BEGIN
          MOVE BUFCHAR := LINECHAR,(1),1;
          @LINECHAR := TOS;
          @BUFCHAR := TOS;
          END;

@LINECHAR := @LINECHAR(1);                          << SKIP CR >>
LEN := LOGICAL(@BUFCHAR)-LOGICAL(@BBUF);
FWRITE(COM'PRINTFILNUM, BUF, -LEN, %40);
IF <> THEN FILE'ERROR;
END;
```

```
SUBROUTINE END'FILE;
BEGIN
FWRITE(COM'PRINTFILNUM, BUF, 0, PAGECTL);    << DO PAGECTL >>
IF <> THEN FILE'ERROR;
IF LOCAL'FILE THEN                    << CLOSE IF OPENED HERE >>
   BEGIN
   FCLOSE(COM'PRINTFILNUM, 0, 0);
   COM'PRINTFILNUM := 0;
   END;
END;

<<
   HERE IS THE MAIN LOGIC OF VPRINTSCREEN
>>

IF COM'STATUS<>0 THEN RETURN;              << JUST LIKE VPLUS >>

START'FILE;                           << OPEN COM'PRINTFILNUM >>
IF COM'STATUS<>0 THEN RETURN;
READ'SCREEN;                          << READ SCREEN CONTENTS >>
IF COM'STATUS<>0 THEN RETURN;

@LINECHAR := @BBUF;

DO PRINT'LINE                        << CLEAN AND PRINT LINES >>
UNTIL COM'STATUS<>0 OR LINECHAR=RS;

END'FILE;                      << CLOSE FILE IF OPENED HERE >>
END;
```

Making The Most Out Of Self-paced Training For Office Software

by
Simon Cintz
Information Networks Divison
Hewlett-Packard

Training people in the use of office software is one of the major stumbling blocks to successful office automation. This paper discusses how self-paced training, which is available for many Hewlett-Packard office software products, can help solve this problem.

Benefits and Costs of Training

Training for office automation has become a major concern for many companies that use computers in the office. If people are trained to make effective use of hardware and software, they can dramatically increase their contribution to their company's success.

The solution to this problem is also well known -

* MORE TRAINING

* BETTER TRAINING

Most people will agree that more and better training means more successful people and therefore a more successful company. So - why aren't companies solving the problem? There are two reasons:

1. TRAINING TAKES RESOURCES. Training must compete with other company needs for necessary resources. Put simply, training often ranks last on a company's list of priorities.

2. TRAINING PROGRAMS ARE POORLY PLANNED. The resources available for training are often

misused and therefore produce ineffective training programs that don't deliver results.

These two problems are often related. Poorly planned training programs often cost the most money. I believe that good training is also efficient training and therefore costs less.

The Solutions - PLANNING and SELF-PACED TRAINING

Since few of us have complete control over the resources our companies will devote to training, we have only one workable alternative -- make sure that the training program we do have is the best that our available money can buy.

This is a planning of resources problem. If you are going to have successful training that's cost effective, you must PLAN your training program. If you are unwilling to spend the time to properly plan your office automation training program, you won't solve the problem -- it'll get worse.

Self-paced training that is currently available for HP office software products can be your ally in planning a successful program. HP has a number of self-paced training materials that are targeted at office personnel. Making effective use of professionally developed and already available training will give you the most for your training dollar.

The remainder of this paper will discuss how to PLAN your training program and make the best use of SELF-PACED materials available from HP.

The Basic Steps

First, let's outline the basic steps to properly plan your office software training program. We'll then discuss how to accomplish each step, with an emphasis on making the most of the self-paced materials that are available from Hewlett-Packard.

1. Identify departmental training needs.

2. Identify audience learning needs.

3. Set departmental training goals/objectives.

4. Identify resources available for training.

5. Outline the training program.

6. Choose appropriate presentation modes.

7. Implement the program.

8. Evaluate and revise the program.

STEP #1 -- Identify departmental training needs

What does your department need to do with office software in order to be successful?

The answer to this question is usually -- "IT DEPENDS".

Find out what IT depends on. Usually, different people need different skills. Be specific - identify what different people need.

For example, if your office has the HPEASYCHART graphics package, find out who uses it and who needs to use it. You may find that managerial personnel need the package to get a quick look at how certain data may be best presented. They are not interested in learning how to operate the plotter to produce charts. They don't want to know about paper pens and transparency pens. On the other hand, clerical personnel may need to know this information so that they can successfully produce a good looking transparency for their boss's presentation.

In identifying your department's training needs, you or someone you designate must become familiar with the office software product for which training is needed. Don't depend completely on current users to know the important product features. Many times current users don't know the range of product features because they have never received training on it. Don't let the blind lead the blind. Have someone take the time to find out how the product features best fit into your department's work environment. If you need help with this task, consider using the consulting services that Hewlett-Packard can provide.

Be thorough in identifying your departmental needs regarding a software product. A training program that starts out with the idea that everyone needs to know

everything about a product is doomed to failure. Trainees will find it a waste of their time. You will find it a waste of your limited resources.

STEP #2 -- Identify audience learning needs

This is a crucial step and is often overlooked. We don't train wordprocessors - we train PEOPLE who do wordprocessing. People have needs in a training program. If you don't address the human aspect of training, you will find it much harder to get the desired results.

There are three major training audiences in office automation:

* PROFESSIONAL - These are the technical business experts. They depend on office software to analyze and represent information.

* MANAGERIAL - These are the decision makers and supervisors. They depend on accurate information and good communications.

* CLERICAL - These are the people that handle the day to day routine of the office. They're efficiency, or lack thereof, greatly impacts office productivity.

These may not be distinct groups and an individual may fall into more than one of these classes.

These audiences have many similarities and differences in their training needs. Here are some of the more important similarities and differences that you may find.

Similarities

1. Most office people are uncomfortable with computers at first. BUT "uncomfortable" is often just a synonym for "untrained". Good training will solve most of this problem.

2. Most office people don't understand computer jargon. Saying a "file is out on disc" is meaningless to someone who doesn't know what a file is or what a disk is.

3. Most people need guidance and structure when learning a new skill. If left to their own devices in learning a new tool with which they feel uncomfortable, many people will be unsuccessful, regardless of how intelligent they may be.

4. Most people like to be recognized and rewarded for their accomplishments. Sometimes these rewards are intrinsic ("I'm doing a better job.") and sometimes these rewards are extrinsic (increased salary or status). Your training program should help people achieve the rewards they desire, whether intrinsic or extrinsic.

Differences
---

1. Managers and professionals are more likely to consider the terminal keyboard a problem. Some of them don't "type".

2. If learning takes place in a group setting, people usually like to be with their peers. When one learns, one usually makes mistakes. Socially, we are better able to deal with our imperfections amidst those of our own social standing. A mistake seems more "foolish" if it is observed by those of higher or lower status than the person making the mistake. Avoid mixing your audiences in group learning situations. There are no hard and fast rules in this area and it all depends on the individual people you are training.

STEP #3 -- Set departmental training goals/objectives

Once you know what your department and audience need, you can set realistic training goals and objectives relative to the software.

The objectives should be departmently oriented. For example,

> "All new department secretaries will begin an HPMAIL training program within their first week of employment."

"All department personnel will be able to send and receive simple messages using HPMAIL."

"Each sectretary will know how to use the 'designate' function in HPMAIL to manage their boss's HPMAIL correspondence."

Naturally, objectives should be specific, observable, and measurable.

Once you've got your objectives together you need to make sure that you have agreement from the people in your department that the objectives are indeed appropriate. Make sure you have the right objectives for your work environment and your audience. It's a waste to have a good training program that teaches the wrong things to the wrong people. Don't make this mistake.

STEP #4 -- Identify resourses available for training

Now that you know what you want to accomplish, you need to find out what resources are available to accomplish the task. Resources can be obtained internally or externally.

Internal resouces include:

** People - instructors, mentors, managers, course developers, etc.

** Money - to hire consults, instructors, purchase materials, etc.

** Equipment - terminals, cpu time, account space, audio-visual materials, classrooms, etc.

** Time - How much time can people spend in training? How much time can you spend designing your training program?

Since individual internal resources vary considerably, it would not be fruitful to discuss their use in detail in this paper. It's enough to say that you should try to get as much as you can and use it wisely.

The major external resource you have is Hewlett-Packard. HP provides some the of the best self-paced training

materials and office support services in the industry today.

Some of these materials are free with the purchase of the software product. Others can be purchased for a few dollars (tutorial manuals) or a few hundred dollars (self-paced training programs).

In addition to self-paced materials, HP also provides classroom courses and consulting services that are designed to help you build a better training program.

Let's take a closer look at some of the resources that HP can provide. If managed properly, they can become the key to a good office training program.

** On-line Interactive Training - this is the least expensive and one of the most effective training tools that HP provides. Some of HP's office products automatically come with this training. For these products, there is no need to purchase separate training materials and everyone who has access to a terminal has access to the training. You don't need a separate copy for each student.

** Tutorial Manuals - these are also low cost items and very effective if used in a structured environment. They take the user through the basic features of a product.

** Reference Manuals - these can be used as part of a training program, but generally do not make good training materials in themselves.

** Self-paced Training Packages - These are complete training programs that usually cost a few hundred dollars each. Don't feel restricted by the word "Self-Paced" -- many of these materials are ideal for use in more structured classroom or tutorial settings.

** Classroom Course (offsite) - HP offers classes at its offices around the world. Classes for office software users are designed to be taken by non-technical people. Often these classes are a good way to train those people who will become your lead office trainers.

** Classroom Course (on-site) - HP will provide an instructor and materials for classes given at your location. This is a good way to train a large number of key people without incurring large travel related expenses.

** Consulting - HP has a worldwide support organization that can provide individual consulting and instruction to your company. These people can help you design your training program and train key people who can then train others.

Not all of these resources are available for all HP office software products. Ask your HP sales representative what is available for the particular office software products in which you are interested.

STEP #5 -- Outline the training program

If you've completed the previous steps, you should be in a good position to outline a training program.

Training programs vary as widely in scope as do the companies that use office software. Some programs involve the training of hundreds of people on many different aspects of office software. Others are targeted at a few individuals who use one or two office products.

I will not attempt to discuss this great variety of possibilities. Instead, I will focus on those programs that make use of the self-paced materials that are available from Hewlett-Packard. These materials have application in a wide variety of situations and can be used by companies large and small.

Use the following steps in outlining your training.

1. Lay out the pieces. You now know:

   * Departmental Training Needs

   * Audience Training Needs

   * Departmental Goals/Objectives

   * Available Resources - Internal and External

2. Take a close look at the self-paced materials that are available for the products on which you wish to train. Better yet, take the training yourself and experience it first hand.

You will find that the training is professionally developed and covers the most commonly used product features. You'll get a feeling for how long it takes to do the training and what, if any, supplementary materials are necessary for your particular department.

Look at the training from the perspective of your departmental goals and objectives.

3. If you have a large audience, segment it. Different people need different training. Here are the more common audience divisions:

   * managerial, professional, clerical

   * some experience with computers vs. no experience

   * new employees vs. existing employees

   * users of one software product vs. another

   Ask yourself who needs what portions of the self-paced training.

4. Look at the training across software products. Every user, regardless of product, may need a simple introduction to the HP3000 computer. Identify those areas of overlap and differentiation.

5. Look at the training across time. Don't try to teach everything to everyone at once. A person needs time to assimilate new information and new skills. Some skills need to be taught immediately, others can wait or be provided on an as needed basis.

6. Take another look at the self-paced training that is available and organize it into an outline that addresses the following questions:

   A. Who will be trained? -- Which job categories will require the training?

   B. What will they learn? -- What topics (skills) will be taught?

C. How will they learn it? -- Which modules of the self-paced materials will be used to teach what? What materials, if any, do you need to develop on your own?

D. In what order will they learn it? -- What skills are most important? Which of these need to precede the learning of other skills?

E. When will they learn it? -- What is the training time line?

Your outline is complete once you have mapped out how the self-paced materials will be used in your training program. An example of a hypothetical training outline for HPMAIL is given in Attachment A at the end of this paper.

6. STEP #6 - Choose appropriate presentation modes

You now have an outline of the training program. All the pieces are identified and fit into your department's goals and objectives. Now you have to decide the best way to deliver the training to your staff. This aspect is dependent on the internal resources that are available and, of course, your needs.

There are three major presentation modes for implementing self-paced training programs:

1. Unmonitored Individual Learning - The student is given the materials and told to learn them on his or her own. No supervision is provided.

   This approach is unlikely to be successful with very many people. Most individuals who are handed a set of materials to learn see this as a secondary task to their other job responsibilities. Most of them will find it difficult to start, much less complete, a training program that provides no guidance or allocates no time specifically for training.

2. Monitored Individual Learning - The student is given the materials along with a written program that specifies what is to be done and when it is to be done. A person is assigned to track the

students progress. An experienced user of the software is available to answer questions. Test materials may or may not be used.

This approach requires more planning than the unmonitored approach, but is much more likely to succeed. The student knows what is expected and when it is expected. An experienced user (mentor) is available to help with problems. Someone is responsible for tracking progress (training supervisor).

Test materials at the end of each unit are desirable but not necessary. Tests can be easily replaced by "do-ables" -- skills that can be demonstrated by some use of the software. For example, after a student has completed the distribution list section of the HPMAIL training, he or she may be asked to send a short message using a distribution list. A copy of the message can be sent to the person who is responsible for monitoring the training program.

This plan usually requires that you develop a short guide or checklist that directs the student through the appropriate materials at the appropriate time.

3. Small Group Tutorial - Students come to a "class" to use the self-paced materials. An instructor is present to direct the use of the materials, answer questions, teach, and evaluate progress.

   This is an excellent way to use the self-paced materials if you have the necessary resources. This method usually requires that someone be an instructor during each training session. It also requires that a small instructional area be set up with the necessary terminals and copies of the self-paced materials.

Choose the implemenation mode that is best suited to your needs and resources. Mix modes if desired. A training program can start out as "Small Group Tutorial" and finish up as "Unmonitored Individual Learning".

Take another look at the various support services that Hewlett-Packard provides. If you need to train specific key instructors or mentors you may wish use HP classes or consulting.

A hypothetical example of a training guide for HPMAIL is provided in Attachment B at the end of this paper. It demonstrates the "Monitored Individual Learning" approach.

STEP #7 - Implement the program

The next step is to implement your training program. I will not discuss this aspect because it varies depending on the program, its size, the company, and the people you are working with. The important thing to remember is that all the key elements of your program must come together at the right time and place - funds, materials, equipment, management support, instructors, mentors, etc..

STEP #8 - Evaluate and revise the program

Once your program is operating you need to begin evaluating its success. Program evaluation should be done relative to the goals and objectives that you first set out. If the objectives are specific, observable, and measureable, you should be able to determine the success of your program.

It's unlikely that your training program will be 100% successful from its first day. Evaluation of your program relative to your objectives will help you identify those procedures that are in need of revision.

Advantages

You will find that using the previously described method for developing your training program has the following advantages:

1. Your program is designed around professionally developed, high quality training materials.

2. Your program is truly suited for your department's needs.

3. You've lowered costs by using already available materials.

4. You've increased people's job satisfaction because you have made it possible for them to be successful with new tools.

5. Your department is more successful because your people are making the most out of hardware and software that is designed to improve their productivity.

Some DOs and DON'Ts

Keep the following list of DOs and DON'Ts in mind while planning and implementing your training program.

* DO tell people why learning a new tool will help them be more successful in their jobs. Sell your trainees on the importance of the product and of the training before they begin learning.

* DO get management support and commitment for your training program. Management can help by providing trainees with the positive incentives they need to learn a new tool.

* DO use successful peers as role models for new trainees. Success can be contagious if the trainees have contact with the right people.

* DO provide on-going technical support after the training program is finished. Trainees need an on-going commitment to their success.

* DO allow people time to take the training. Allocate the necessary time for the trainee to do the training without interruptions from his or her normal work activities.

* DON'T rush the trainees by trying to do too much in too little time. It is better to do a few things well and slowly than to do everything too fast and poorly.

* DON'T be too quick to blame your trainees for a less than successful program. If your program is having problems, review the program throughly. Blaming the trainees is not the solution. Change what you can; change the program.

A final word

Office personnel have recently received much "bad press" regarding office automation. Many times the slower-than-desired pace of office automation is blamed on people who are "slaves to old ways" and "afraid of the computer".

I believe this is a great overstatement of blame. It is true that people are slow to change but is also true that people are very accepting of change if it is perceived as BENEFICIAL and ACHIEVABLE. Much of the blame for the slow pace toward office automation falls on the shoulders of those who fail to provide proper training, or worse yet, no training at all.

Most people need assistance in learning new skills. Our failure to get people to use the computer as a tool is often because we focus so much on the machine that we forget the person using the machine. People need training. They need good training. It is our responsibility to make sure we provide this vehicle for their success.

Attachment A

HYPOTHETICAL TRAINING OUTLINE FOR HPMAIL

Audience: Clerical

Necessary Materials:
* HPMAIL Interactive Training (on-line)
* HPMAIL Reference Guide (booklet)

Prerequistes:
* Simple familarity with the HP3000 (The Guided Tour - Module 1)
* Simple familarity with the terminal (The Guided Tour - Module 1)

PHASE I - Introduction

(approximate time: 2 hours)

Topic: What is HPMAIL and why is it important in this company.

Materials: None. Experienced user discusses HPMAIL with trainee and gives short demo.

Topic: Sending and Receiving Simple Messages

Materials: Module 1 of HPMAIL Interactive Training

Topic: Setting Passwords

Materials: Module 5 of HPMAIL Interactive Training

Topic: Using Distribution List Functions

Materials: Module 2 of HPMAIL Interactive Training

PHASE II - Editing and Filing Documents

(To be done one or two weeks after PHASE I
is completed. Approx. Time: 1 hour.)

Topic: Editing Text Documents

Materials: Module 4 of HPMAIL Interactive Training
(If already familiar with terminal
editing keys, only Lesson 2 of this
module is necessary.)

Topic: Filing Documents

Materials: Module 3 of HPMAIL Interactive Training

PHASE III - Tracking and Copying Documents

(To be done one to two weeks after PHASE II
is completed. Approx. Time: 1 hour.)

Topic: Tracking Documents

Materials: Module 6 of HPMAIL Interactive Training

Topic: Copying Documents

Materials: Lessons 1 and 2 of Module 7 of HPMAIL
Interactive Training

PHASE IV (optional) - Advanced Functions

(These topics can be done on an as needed basis.)

Topic: Handling MPE files

Materials: Lesson 3 of Module 7 of HPMAIL
Interactive Training

Topic: Sending Packages

Materials: Lesson 4 of Module 7 of HPMAIL
Interactive Training

Topic: Autoanswer and Autoforward Functions

Materials: Section 8 of HPMAIL Reference
Guide

Attachment B

The following is an hypothetical training guide for PHASE I of the previously outlined HPMAIL training program. The audience is clerical personnel and the mode of presentation is "Monitored Individual Instruction".

HPMAIL Training Guide
PHASE I
****

Trainee Name:_____ Ext:_____

Department:_____ Supervisor:_____

Is Trainee registered as a HPMAIL user?_____

Have prerequistes been completed? _____

Date Phase I training started:_____

Date Phase I completed:_____

If you have questions about the training, call:_____
_____ (Training Supervisor)

If you have questions about using HPMAIL, call:_____
_____ (HPMAIL mentor)

The following is a training guide for learning HPMAIL which is our office's electronic mail system. There are four phases to the training program:

1. Introduction (approx. 2 hours)
2. Editing and Filing Documents (approx. 1 hour)
3. Tracking and Copying Documents (approx. 1 hour)
4. Advanced Functions (optional)

This guide will take you through the first phase of the program - the Introduction. Please follow and complete each step as directed.

___✓___ Check off each item as you complete it.

STEP #1 (10 - 15 minutes)

____ Go see your HPMAIL mentor for a brief introduction to HPMAIL. He/she will show you how to run the Interactive Training.

____ Write down the command for running the training:_____

STEP #2 (30 - 40 minutes)

____ Run the training at your terminal. Do lesson 1 in module 1.

____ Exit the training program.

____ Sign-on to HPMAIL using your own name and read the message(s) you have.

____ Exit the HPMAIL program.

(If you have problems, call your HPMAIL mentor for help.)

STEP #3 (30 - 40 minutes)

____ Run the training again. Do lesson 2 in module 1.

____ Read the Summary in module 1. Get a paper copy of the Summarry at the printer.

____ Exit the training program.

____ Sign-on to HPMAIL and send your Training Supervisor a short messsage.

____ Print a copy of one of your IN TRAY messages and attach it to this training guide.

____ Exit the HPMAIL program.

STEP #4 (5 - 10 minutes)

____ Run the training again. Do module 5.

____ Exit the training program.

____ Sign-on to HPMAIL and set up a password for your mail. IMPORTANT - Don't forget your password!!!

____ Exit the HPMAIL program.

____ Sign-on to HPMAIL again. You'll be asked for your password.

____ Exit the HPMAIL program.

STEP #5 (20 - 30 minutes)

____ Run the training again. Do all of module 2.

____ Exit the training program.

____ Sign-on to HPMAIL.

____ Set up a short distribution list, include your Training Supervisor on it.

____ Send a short message to this new distribution list.

____ Exit the HPMAIL program.

____ Call up your HPMAIL mentor and ask him/her what distribution lists are used in the company.

STEP #6

You have now completed PHASE I of your HPMAIL training.

Take this completed guide to your Training Supervisor.

Security issues: How secure is YOUR system?

Doug Claar, programmer analyst
Hewlett Packard, Computer Systems Div.

Many new computer users implicitly expect that the computer which they are using is private and secure. System managers understand that this is a fallacious assumption, as they are able to access everything on the system. What system managers often do not understand is that they are not always the only ones with access to the entire system. Unless a system manager consistently stays on top of system security, that security quickly evaporates, but if typical security breaching techniques are known and understood, appropriate steps can be taken to foil takeover attempts and restore secure status to the system. It is essential, then, to first understand what a system invader has to go through in order to take over a system to be able to effectively combat security violations. Reformed burglars are said to make the best security experts, and the same might be expected of computer security experts: those who have broken into systems are best able to specify the countermeasures that would have worked against them.

Before discussing some typical security breaching techniques, a disclaimer is in order: neither the author nor Hewlett Packard (especially Hewlett Packard!) condone unauthorized access of computer systems or of any data thereon. The breakin techniques are described strictly in order to discuss the appropriate counter-measures.

To better understand the techniques described later, it is helpful to present the EDP environment in the Computer Systems (CSY) R&D lab: Although the attitude is perhaps not universal throughout Hewlett Packard, in the Computer Systems (CSY) R&D area everyone is encouraged to utilize the computers as much as possible. Employees using the 3000 (even for personal projects) inevitably benefit the company. People are allowed use the 3000 (to write papers, do homework or whatever) on their own time, and are encouraged to look for ways to use it on the job. HP benefits from more sophisticated users, and from the programs written by those users. It is a natural outgrowth of this attitude to not have strict security, but as more sensitive information makes its way onto the lab 3000s, this attitude is changing.

A 3000 can be as secure or as insecure as it's management desires. In the past it was argued that lab systems should be low in security, since the machines were strictly R&D--no accounting, payroll, or management functions. It is only recently that the realization has begun to dawn that security is as important for us as it is for our customers. The installation of about fifty dial-in lines has been the impetus for tightening security on the R&D timeshare machines. In addition, as CSY's computer literacy push bears fruit, more people are using the systems for more sensitive data. As an example, several secretaries are beginning to type employee evaluations on the system. EDP has assumed the responsibility of providing the greater degree of security that such users require and expect and of educating them about any requirements placed on them for security of their data. The job is not nearly complete (especially in the area of user re-education), but work is progressing steadily on providing a secure environment for all users.

Although most lab systems have traditionally had low security, there have always been one or two secure labs systems: those whose system manager or other responsible person was individually concerned with security. The system managers on these systems have made it their job to try to get past each other's security schemes, and most of the techniques covered come from that source. Today, their solutions are finally beginning to be put into use lab-wide.

Security solutions must deal with the question of what system security is. From the unauthorized user's point of view, system security is "what is in the way". To get around security, this user must accomplish three key objectives, which are to get onto the system, to work into a position of power, and finally, to leave the pieces in place to facilitate repeated re-entry. From the System Manager's point of view, security must consist of making each of these objectives as difficult as possible to attain. Let's look at some techniques for achieving the unauthorized user's objectives, and then at ways to block these techniques.

The first requirement for getting "into" a system is to get "onto" it. There are several potential weak spots, with perhaps the most obvious being facility, company or MPE common user IDs. For example, at CSY, almost every system has, in addition to standard MPE IDs, a DS user ID (for DSing through to another system), an I/O utility ID (for transferring spool files to a system with an EPOC on it), and an electronic mail ID (for remote HPMAIL users). It is likely that many if not most computing centers also have some type of common user ID--perhaps even for demonstrations. Although these accounts may not have any special abilities, they provide a foothold (or beach-head) from which to launch an assault.

A second potential weak spot is user IDs from adjacent systems, especially if those systems are DSed to the target system. Often, legitimate users will have the same user IDs and passwords on adjoining systems for their own convenience. Assuming the user's ID and password can be discovered on one system, the adjacent system can also be penetrated.

Should both of the aforementioned methods prove fruitless, there are still other techniques available to the determined intruder. One time-honored technique is to look for user IDs among discarded listings or at unattended terminals. An example of this is when one of the college students we hired for the summer came back the following summer and found all his capabilities gone and his terminal hooked to one of the more secure lab 3000s. After challenging him to get back the capabilities, I was called away from my desk. While I was gone, he simply walked over and upped his capabilities from my terminal. The moral is: people are the weakest link in any security scheme. User carelessness must be combatted by education as well as top level techniques. Users, especially those who require special capabilities, must be convinced that leaving terminals logged on is like leaving a car running, and should be accompanied by close supervision. (If connect time is billed, this point is probably easier to make).

Because any user must first get onto the system in order to do anything, an obvious first administrative step in combatting unauthorized use is to limit access to only those who should be on the system. There are several things that can be done to limit access: eliminating or severely limiting common user IDs is a good first move. For example, the three common CSY user IDs mentioned earlier can all be restricted so that they are able perform no other function than their intended one. Looking at a specific example, it can be seen that, in the case of the DS ID (which is intended only for people using DS to get to another system on which they have an account), that systems which are at the end of a DS line do not need this ID, and that those which do require it could limit it to DSLINE, REMOTE, and perhaps FILE commands. The limitations can be accomplished with a UDC that aliases all other MPE commands to either a no-op or a logoff, with operator and system log indentification. (If CONSOLE logging is turned on, a simple TELLOP will notify the operator and be recorded in the log).

Other common logons can be analyzed and controlled in the same manner: decide what the logon was designed to accomplish, and disallow everything else. Be aware, however, that some programs and subsystems, such as TDP, allow the user almost full MPE functionallity without the restrictions of UDCs. These programs are more troublesome to control, and should thus be disallowed to common (transient) users whenever possible. In general, it is not a good idea to allow

any form of the run command in this environment: In the case of electronic mail at CSY, having users type "HPMAIL", a UDC in which the break key won't work, is much simpler--and safer--then having them type "Run HPMAIL.HPMAIL.SYS".

MPE common users need not be a problem either, even if operators are not on duty at all times. At CSY, all .SYS users have a UDC which prevents their use from any location other than certain specified terminals. In addition, the Operator.Sys UDC disallows STORE, RESTORE and many other functions (including SETCATALOG!). Because engineers must be able to bring a system back up on the weekend (instructions are posted on each system), Operator.Sys has no password: its home group has, however, been changed to LOGON, a group created with virtually no capabilities. Because passwords tend to "leak" out, the passwords for Manager.Sys and the pub.sys group are changed frequently. To facilitate this, a program has been written that changes the password, if any, contained in the first record of a group of files specified by the program user. Surprisingly fast, this program makes password changing much less traumatic and time consuming. (This program, named NEWPASS, will be available on the swap tape, but--like everything on the tape--it is not guaranteed).

Assuming that an intruder has been able to log onto the system, their next objective is to move to a position of power. There is no (known) way to bypass security using only standard user capabilities, but that does not mean that the person breaking in needs higher capabilities: only that someone else has "left the keys in the ignition". In fact, the ideal situation for the interloper is to find, or leave behind, a "superman" program (one that gives "super" capabilities) in some innocuous place, and then in the future only log on as a mild-mannered, common user. To plant such a program, (if one can't be found) what capabilities are needed? Obviously, either System Manager (SM) or Privileged Mode (PM) would work quite nicely, but since those capabilities are usually guarded very well, what else might help? System Supervisor (OP) capability will also work, since a user with this capability can restore any file anywhere and can also dump the account structure. Account Manager is useful only if the account has OP (system supervisor), SM (system manager) or PM (privileged mode). A .sys logon is useful because files can be restored into .pub from any user.sys, even with only standard capabilities. By the same token, any user within an account can restore to any group in that account, allowing non-privileged users to restore a file to (someone else's) privileged group. Of course, the other user will wonder where the file came from, so it is a good idea eventually leave the program in a group with loads of files, or in pub.sys with a name like "HIOCARD2".

There are several ways to conduct the search for power. The most obvious (and usually most fruitful) begins with a listf and a knowlege of what people tend to call things. It is amazing the number of people who will call a capability program "CAPS", "GETCAPS", "SM", "SUPERMAN", "PRIVS", or the like. In addition, people tend to identify stream jobs with a J or S, and since MPE requires the passwords be in the file, access to stream job files can be very "helpful" to the intruder. A third group of "useful" files that tend to be named similiarly are UDC files, although these file names can often be found more directly from command.pub.sys. The key point is that meaningful file names can be a two-edged sword: both users and abusers can benefit. Programs that store user and account information on disk are especially dangerous: meaningful file names here can be disasterous if the program accidently (or purposely) leaves the files behind. For example, a programmer at CSY wrote a utility program that read the account structure from one 3000 and formatted it onto a stream tape so that another 3000 could have the same structure. The program worked fine, with one minor flaw: it left three files--TACCT, TUSER and TGROUP--on PUB.SYS with account, user and group information (including passwords) in them. This program made it all the way to Boise and Fort Collins before its "feature" was realized. Taking advantage of mnemonic names is simply one example of a way to get into a position of power. There are doubtless many others.

System managers must of course be responsible for their own logon, but also ultimately much more: the entire system. The System Manager must administer all data pertaining to the system, all access paths to the system, and all capabilities on the system. The most critical data pertaining to the system can be found on the SYSDUMP tapes: a complete sysdump tape set <u>is</u> the system--in terms of everything but physical hardware (which plant security hopefully monitors). Sysdump tapes should not be available to general public: they should be locked up and, if a file needs restoring, EDP should do it.

The access paths to the system should also be controlled as much as possible. Although hardwired terminals may require monitoring, phone and DS lines are probably more of a concern. Analysis of these paths should include the identification of who uses them, and why. If, for example, a DS line's purpose is to allow the users of one system to access a central resource, then the DS line should be made one-way by eliminating the virtual terminals DS users need in order to log onto the system. If there is occasional two way access, then steps should be taken to insure that communication is limited to those who should be using the line.

System logging, along with some type of data formatter to print appropriate parts of the log, can be used to monitor

those who log on to either phone or DS lines. There are several contributed library programs that crunch log files, and if those aren't suitable, the system manager manual provides log file format information for do-it- yourselfers. (A rather inelegant but simple program used on one of the CSY systems will again be available on the swap tape as LISTLOGF (with the standard "no support" proviso).

To control those who log on by either phone or DS lines requires some way of knowing what LDEV is being used. A popular way to do this is to set up a UDC that executes every time anyone logs on. This logon UDC, which should not allow the user to break, or to see the UDC definition, can simply execute a security program and log the potential user off if all is not kosher. Because a program has access to MPE intrinsics, it can determine if the user is coming from an LDEV that is defined as DS or dial-up, and can then ask for a password, or just deny access altogether. Besides testing for phone or DS lines, the program can also test for many other conditions: CSY's program also tests for Operator.Sys and LDEV 20 (they must occur together). Once again, this program (and associated UDC) will be made available on the swap tape as "STARTUP".

Finally, capabilities of legitimate users of the system must be monitored and controlled, as those users will also often see how far they can get on their own system. Thus, after dealing with the outside world, it is time to look inward at protecting users from each other and themselves. Although this is an area in which most system managers have much expertise, it will not hurt to point out several things to watch for. If any users with privileged mode are allowed, they should reside in an account separate from non-privileged users, with user, group and account passwords. Treat privileged mode as if it were radioactive or highly explosive--it is! Remember that OP capability allows unlimited store, restore and sysdump capability. Also, why create .SYS users who can then restore into PUB.SYS? There has to be another place for that user to go. Be constantly on the lookout for new privileged mode programs, user, groups or accounts. Use the list of standard MPE files provided in the communicator to verify which files should be in the SYS account. Use LISTDIR2 to verify that LISTDIR2 has not been released: secure it if it has.

Two programs used at CSY to keep tabs on on privileged mode are included on the swap tape. Neither are fantastically elegant, but they both work. The first, LISTFPM, looks for files that require PM capability to run. At CSY, we stream a job, included on the swap tape as LISTFPM.JOB, which runs LISTFPM, lists the secure/released status of LISTDIR2, and runs the log file analyzer program (LISTLOGF). The second program, LISTUSRS, prints a formatted listing of pertinent user and account information, with or

without passwords. This program and its output are as dangerous as dynamite, should be handled accordingly. Don't leave the listing on the printer, in a spool file, or on a desk. Don't even throw it away without shredding--remember, people do look through discarded listings. If the listing is left unattended, even while waiting in a spooler queue, someone might copy it. And no one really wants the visibility of having to tell users that they have to change all their passwords because EDP blew it, or the chore of changing all the passwords that EDP is responsible for.

It takes some time and effort to ensure a secure system, but thankfully, there are tools available to help do the job. Although no computer system can ever be one hundred percent secured, the steps outlined here should make the security fence high enough to keep the vast majority of trouble-makers at bay, to trip up the few that get by, and to give users the level of protection they want from all computers.

(The author is interested in exchanging security ideas, horror stories, problems etc. with other 3000 users).

SUPPLY SIDE PROGRAMMING
(Parts I and II)

by TIPTON COLE

Technical people must learn to work on business, rather than technical, solutions to business problems. Because of our technical tunnel vision, we spend too much time and money developing software that does not serve the business. For any business problem there is a solution that answers the need at the lowest possible cost and in a reasonable amount of time. The conscious pursuit of this solution is "global optimization."

The four key ingredients for global optimization are: 1) work on the right problem, 2) optimize only where it helps, 3) optimize for the lifetime cost of the project, and 4) use proven methods. This paper introduces each of these four key ingredients and expands on the first. Later papers will explain the others more completely.

Work on the Right Problem.

Your chance of success improves dramatically when you are working on the right problem. You can't determine what the business needs most without asking the right questions of the right people.

Optimize Only Where It Helps.

This point is particularly important for programmers who spend a lot of time writing "efficient" code. We don't know where the optimization will help until we have measured the program performance.

A good example of our misdirected pursuit of efficiency was the two-year quest to eliminate disc accesses. After reading Bob Green's paper on optimizing on-line systems in the 1978 IUG Proceedings of the Denver Meeting, we immediately sought out and eliminated disc accesses whereever possible. Minimizing disc accesses on the HP3000 became the watchword of "knowledgeable" programmers. After all, nobody knew better than Bob Green, and he had said that reducing disc accesses was the way to optimize your on-line programs.

At the 1980 IUG Orlando Meeting Bob presented a paper on optimizing BATCH programs. This was a bit of a surprise for most of us. We already knew how to optimize programs - minimize the number of disc accesses. But Bob had actually run the tests and made the measurements to determine the actual bottleneck in batch programs. CPU utilization. Not disc. CPU. As one we replied, "Oh."

Almost all of the time that we had spent trimming disc accesses in our batch programs for the last two years had been wasted. It may have helped a little bit, but certainly not enough to pay for all of our effort. Bob had not led us wrong. Beginning with the title of the first paper, all the way through with qualifications and caveats, he told exactly what he meant, exactly what he had measured. But we thought we knew much more than he had told us.

We spent a lot of time "optimizing" batch programs without any benefit to our employers or customers.

Optimize for the Lifetime Cost of the System.

HP terminals. How many of us purchased Brand 'X' terminals because HP terminals are too expensive? How many of us wrote our own screen handlers because VIEW-V/3000-VPLUS was

"inefficient" or because the terminals are too expensive? We forgot that over the long haul our real investment is in software. HP knows that. Their commitment to preserving software compatibility is one of the best reasons for choosing them as hardware suppliers.

So we buy cheap terminals and write our own screen handler. Later one of our system level tricks catches up to us, and we can't use our screen handler with the new operating system. So we have to plough into the old programs to find out how the screen handler works. The author, of course, is long gone. Since he was concerned with efficiency, including such things as disc space, his program doesn't have SPL statements on separate lines. The screen handler is, in effect, one long string.

By the time that this system is retired, measured at the end of several years, those $800 terminals have cost us about $6000 each.

Use Proven Methods.

Look around you. Somewhere someone is doing a good job. Maybe not in your city, but just maybe in your state. When you find that person, DO IT THE WAY HE DOES IT. Whoever you find, however he makes it all work, don't waste any time getting in line to follow the leader.

If he writes exclusively in RPG, but he is getting through 40,000 transactions a day on his Series III with 256K of memory using 40 VIEW terminals, do it his way. Until you find someone who does it better, or you can PROVE that you have developed a better way.

We can't just rest on our current technical skills or protect our ego or vent our biases. Our job is to get results. To get the results in a reasonable time at a reasonable cost.

WORK ON THE RIGHT PROBLEM

There are two immediate impediments to working on the right problem. First, the users don't know what they want. This shouldn't be a surprise to anyone, and it not really a joke. They don't know what they want largely because they don't know what is available. In many respects WE don't know what is available either.

Our job is not always easy. It is scarey picking out $60,000 worth of terminals, realizing that, unless you've been talking to the right guy in the HP labs, your terminals may be "obsolete" in six months. But by considering the acquisition as a business decision, some of this anxiety can be relieved.

Remember the guy who puts off buying a computer because, "I can get it cheaper next year." Somewhere along the line, he has to buy, or he will never get the benefits of having a computer. At some point, regardless of possible future price reductions, it will pay him handsomely to buy the computer now, just to get today's job done.

While he is waiting for the best possible price/performance on the hardware, his real problem goes begging. His business suffers.

Our job is to work with the users to make the right business decision, taking into account the urgency of the need, the effectiveness of existing systems, the suitability of existing hardware and software, cost, and our ability to put together all of the pieces.

The users don't know what they want. What is worse, though, is that they THINK they know what they want. How many times have you sat down with a user, and his opening remark is something like, "I want you to develop an order entry system for me." And you say, saluting as you leave, "Yes, sir!" You march back into the cubicles, and six months later you return with his brand, spanking new order entry system. Have you solved his problem? Do you even know what his problem is? All you have done is give him the solution that he asked for.

Computer programmers are, by nature, problem solvers. Most of us are eager to please, though many of us are cured of that after a few years. But when we start out we are eager to please, and we really work at what we are asked to do. We should learn to ask, "Why? Just what is it you expect to accomplish with this order entry system?" And thereby hangs a tale.

True Confessions.

We once had a customer. We were their third software supplier in about two years. On our first visit, they told us, "We want you to write an order entry system for us." We spent the rest of the meeting talking about how to do it.

After six months of design, programming, and meetings, we were ready with the order entry system that met all of their requirements. But when they saw it run, they said, "It doesn't look right, and anyway, we've changed our minds about a few things."

So we went back and wrote another order entry system. When we brought that one up, they said, "Gee, that's just what we asked for, but, you understand, our inside salesmen don't really like working with computers, so..."

Finally we asked the magic question, "Why? Just what is it you expect to accomplish with this order entry system?" And they said, "We need to get our invoices out faster. From the time that our goods leave the dock, it takes us three weeks to get out an invoice. The money we borrow to cover that period costs us about $250,000 a year in interest. Frankly, we'd rather have that $250,000 than pay it to the bank."

Well, that put an entirely different complexion on things. They already had an inventory system. Within the inventory system they were already collecting 60-70% of the information needed to write invoices. All they REALLY needed was a back-end to the inventory system to put in the remaining 30-40% of the information needed for invoicing. That is a very, very different job from writing an order entry system.

Our customer had heard that one benefit of an order entry system was that it printed invoices quickly, as soon as the goods were shipped. He wanted his invoices quickly, so order entry must be the ticket. He had given us the solution rather than the problem. He thought he knew what he wanted.

But his only serious mistake was relying on us. We were the ones who failed. It is our responsibility to find out what the real problem is. The user knows what it is, and the only way to get it out of him is to ask the right questions. He cannot possibly know how to express his needs in computer/technical terms. But if we ask the right kind of questions, he will tell us exactly what he needs and how much it is worth.

The Fast Track.

You get some very nice benefits from asking the right questions. Imagine the reaction the first time you sit down with one of the vice presidents and ask him, "Just what is the return you expect from this system? What is the net present value of the savings from this system? Given the

risk, is it worth the investment?" Suddenly you're on the fast track. You'll be a manager in two months. You're talking business to him, and he understands you.

Be careful; don't go too far. When you're really rolling with questions on rate of return and cost/benefit analysis, and you ask, "Is it important to make a profit in this business?" you'll find yourself with a different set of duties entirely.

Remember also that in almost all businesses data processing is a cost center. You're part of the overhead. Users want to get their work done. Business things, like payroll, inventory, payables and receivables. They aren't in the structured programming business or the data base business or the block mode business or the COBOL business. They don't care about subroutines or USL's. They just want their stuff to work. The sooner, the better. But it better work.

In order to give them what they really want, though, we have to translate their needs and wants into structured COBOL programs that use the data base and the block mode screen handler. We write subroutines and PREP USL's to get the final result.

To make a good translation, we have to ask the right questions.

Look Beyond the Computer.

Working on the right problem often means NOT using the computer. Figure 1 reveals a very common mistake: working on a computer solution for which there is no corresponding business problem. We may be learning a lot about the internals of MPE, or we may be learning just how to get past the snake, but these "computer solutions" may not be paying off for the business.

Looking beyond the computer doesn't mean trying to solve every problem with 3-by-5 cards and ledger books. It does mean keeping your sights set on the business problem.

Take the terminal selection problem. Who gets assigned to select terminals? The guy who wrote the screen handler? After all, he's the one who knows the most about terminals. But is this really a technical decision? What you need to look at is the cost, from the time this project begins until the time it is replaced, all through its life. How much does it cost?

Whatever the problem that data processing is called on to solve, selection of terminals, developing a new system, buying any software or hardware, we must address ourselve to the underlying business needs to find the best solution.

Down time, software, support, obsolescence, money, opportunity, efficiency. How much does it cost? Convert it all to dollars, then compare to the cost of doing it other ways. These are hard questions to answer. They require technical information and informed technical judgement, but they are business questions that require business solutions.

OPTIMIZE ONLY WHERE IT HELPS

Programmers usually want to use the newest, fastest features of the hardware and software. We want to use SPL because it generates the most "efficient" code on the HP3000. This in spite of the well-known feature that a program spends 90-95% of its time in only five or ten percent of its code.

Until the most used 5% of the program is optimized, any effort spent on the rest of the code is wasted. It does no good to make the rest of the program go faster when it is hardly ever executed. We cannot justify several days of a programmer's time spent to double the speed of a piece of code that runs for only 9 seconds a day.

The hard part of optimizing any program is knowing where the program is spending its time, finding the critical 5%. Donald Knuth is the author of the series "The Art of Computer Programming." He is, I believe, one of those rare people who can read a book or an article on a subject and understand it completely. The rest of us have to stub our toes a few times before the lesson is really brought home. Donald Knuth says that he cannot read a program and tell where it can be optimized. If he can't, then the rest of us mortals don't have a chance. Therefore, the only way to know with assurance that our efforts at optimization are directed at the right code is to measure the program performance BEFORE diving in. There is not other way to do the job right.

A Cautionary Tale

Large parts of Cole & Van Sickle's PROTOS product are written in Cobol. We provide executable modules compiled using both the COBOL and COBOLII compilers. Because we want to maintain only one version of the source code, we did our string manipulation using indexes and arrays to test and move individual bytes.

In one of our earlier efforts to make PROTOS more efficient, we replaced this string manipulation code with an SPL routine that uses the SCAN and MOVE instructions. On typical test data the COBOL-compiled version of PROTOS ran about 20% faster with the new SPL routine, but the COBOLII-compiled version ran 4% slower with the SPL than with straight Cobol.

This experience taught us two things. First, the COBOLII compiler generates some very efficient code. Second, our assumptions about how to write efficient programs need to be reexamined on occasion.

Measuring program performance can be a tedious and time-consuming task. We used the LOOK/3000 package from Wick Hill Associates to pinpoint the most executed code in PROTOS. With that information we were able to concentrate our optimization efforts where they would do the most good. Other products which may be useful are OPT/3000 and APS.

The 80-20 Rule

There is a general rule of business called the 80-20 Rule. The 80-20 Rule says that you get 80% of your business from 20% of your customers. You get 80% of your trouble from 20% of your customers, too. Usually not the same 20%. Did you ever wonder why HP concentrates on the major accounts?

It makes good business sense to take advantage of the leverage of the 80-20 Rule. Within the business community it goes without saying that new projects are taken on or new products are added only if the benefits to the business outweigh the costs. Often features are trimmed from a product to keep production costs in line with the pre-determined selling price. By the same token, the addition of few features can turn an ordinary product into a luxury item that brings a much higher price. In each of these cases the effect of the 80-20 Rule is apparent:

> Most of the features are available at relatively little cost, but the "finishing touches" are very expensive.

It may seem too simple, but the 80-20 Rule is a basic operational principle of business.

Extended to programming, the 80-20 Rule says that we get 80% of our results from 20% of our effort. Stated another way, it says that 80% of the job is easy. Failure to recognize this phenomenon really hurts.

We reach the point in every project where we have the main part of the project completed. Then we make the mistake of projecting linearly. We have done 80% of the project in four weeks, so the remainder can be finished in one week. We may even pad it out and say two weeks. Years of experience and the 80-20 Rule tell us that this linear projection is wrong. The great majority of the work remains to be done.

This explains a lot of missed schedules and overrun budgets, not to mention untold frustration and self-doubt. We, our user, our customers and our companies are victims of this misguided linear assumption.

We can make the 80-20 Rule work in our favor, though, if we go back to the basic cost/benefit principle of business. Sometimes the easy 80% is all that is really important. A good example is in teaching programming. Remember the programs that you wrote back in school. When you wrote an operating system or an accounts payable system, you didn't produce a commercially usable product. The purpose was only to teach you the general principles involved. That served your needs for the time being.

We often neglect the importance of time when we select the next task to work on. We focus on technical solutions and technical completeness instead of business utility. Serious and costly business needs go begging while we decide whether to add a sort field or while we reprogram existing systems unnecessarily.

Judicious selection of features can make a business application available this month rather than the end of the next quarter. Since time is often so valuable, we owe it to our employers to present all of the options of features, costs and time so that they can make an informed selection. When we know what is most important, we will spend our efforts on the productive work.

The 80-20 Rule matches up very well with the rule mentioned earlier about where programs spend all of their time. These rules tell us to evaluate goals and options before we start to work so that we can direct our efforts where they will be most useful.

The 80-20 Rule: Second Application

Assume that we have made our first pass at the project, so we have implemented 80% of the functionality with a mere 20% of the effort needed for the whole project. Apply the 80-20 Rule to what remains:

80% of the remaining functionality is

```
.80 * (1.00 - .80) =
.80 * .20 =
.16 = 16% of the total project
```

20% of the remaining effort is

```
.20 * (1.00 - .20) =
.20 * .80 =
.16 = 16% of the total effort
```

Added to the results from the first pass at the project, the second application of the 80-20 Rule tells us that we will get 96% of our results for 36% of our effort.

Pause for a moment to let this sink in. This is a dramatic illustration of the value of concentrating on the parts of the system that yield the best results for the least effort.

VITA

Tipton Cole has worked in the HP community since 1977. He speaks frequently at international, national, regional and local user groups on the topic of global optimization and step by step system development. He is an invited speaker on HP's Productivity '83 tour. Formerly a vice president of the Cole & Van Sickle Company, he now has his own software development and consulting business in Austin, Texas. Tipton Cole is a graduate of the University of Texas at Austin with degrees in math, computer science, and law.

SIGNING THE DOTTED LINE
An Updated Perspective of Computer Law

Authored by:
Melissa J. Collins
Manager, Information Processing
Kutak Rock & Huie
The Omaha Building
1650 Farnam
Omaha, Nebraska 68102 USA

INTRODUCTION

By the end of 1985, it is estimated that there will be an many as one and one-half million computers in use in the United States. Viewed once as merely a tool for large bureaucracies such as the census bureau, the computer today has "come of age" in the business community, where almost $60 billion will be spent for computer-related goods and services in 1983.

Although the enormous size of this industry is enough to indicate the number and variety of problems likely to be encountered when trying to legislate and regulate an ever growing industry, the problems are aggravated by the fact that the industry has been in existence for less than thirty years. It should come as no surprise that just as the computer has brought about revolutionary changes in certain segments of society, courts find themselves facing equally new and complex legal issues in areas such as the negligent use of computers, the right to keep computer records private, the ability to copyright and contract grievances.

This article makes a brief pass at all pertinent issues arising in the arena of computer law today. Because of its brevity it should be used as a guideline for further research into the areas the reader should possibly explore. Much of the content of this paper was drawn from readily available sources such as the Computer Law and Tax Report, the Harvard Law Journal, the Computer/Law Journal and several articles in technical periodicals.

CONTRACTS

Contractual disputes involving computer-related goods fit nicely into two distinct groups; hardware, including the computer itself and its peripheral equipment, and software, including computer software, including computer programs of all types, together with the products of the computer systems design effort (e.g., flow charts, documentation, etc.). Though the following discussion will concern mainly computer hardware and software supplied for commercial purposes, many of the same legal principles apply equally to the ever-increasing home computer market.

The term "computer system" will be used to refer to computer hardware and/or computer software. This is done not only for ease of discussion, but also because computer hardware and software are increasingly being sold or leased as a unit intended to produce a specific result. Where applicable, however, and especially when discussing implied warranties, computer software will be considered separately because of the special problems presented by the Uniform Commercial Code (UCC). This discussion does not provide a checklist of items to be considered when entering into contract negotiations for computer systems, but concentrates on the issues of liability which arise after the agreement has been entered into.

Many cases arise out of a basic disagreement over what was to be supplied in the computer system. Many time these problems arise from the purchaser's inability to request exacty what he requires. When the results achieved by the computer system do not live up to these expectations, it is natural to consider those who supplied the machine as the cause of the difficulties.

The paragraphs that follow are organized around the issues which have been most frequently raised in the reported cases involving contract disputes for computer goods and services. The issues have been arranged according to when they would arise during the contractual relationship. The discussion, therefore, begins with an examination of statements made prior to the signing of any actual contract, to determine if these statements were sufficient to constitute fraud, or if they were excluded from the agreement by operation of the parol evidence rule. Next, the agreement itself is examined to determine the outcome of issues involving express and implied warranties and limitations of liability and liquidated damages clauses. The remedies which the courts have found appropriate are evaluated along with an examination of possible damages.

Finally, the article discusses the special problems presented in the case of the assignment of rights under a computer lease.

Representations

Despite their technical sophistication, the same basic marketing strategies are usually followed in selling computer systems as with other large ticket items. What is said and done during the early stages of this marketing activity can have profound impact on subsequent questions of legal liability. When a computer system that is leased or purchased does not live up what the purchaser believes promised, the first question to consider is whether the contract was signed in reliance on a misrepresentation by the vendor. Courts permit some "puffing" by salesmen holding that it is not reasonable to rely upon what is basically a personal opinion. However, the line between representations of fact and opinions is often very narrow.

There are several elements of misrepresentation which the courts have dealt in the area of computer systems. First, if the purchaser knew or had reason to know that the representation was false, the claim will fail. Next, reliance by the purchaser on the representations of the vendor must be reliable. Finally, the purchaser may not have relied upon the misrepresentaion made by the vendor at all (for example the case of Investors Premium Corporation vs. Burroughs Corporation).

An area of special importance in misrepresentations involving computer systems is the relative knowledge of computers held by the parties.

Warranties

Absent a finding of fraud, and assuming that the vendor is successful in excluding promises made prior to the signing of the written contract, the vendor may still have breached an express or implied warranty. Most express warranties in contracts for computer systems are limited to the replacement of defective parts and the correction of errors discovered in the software. As with a contract for any item, the courts will uphold valid express warranties, but will also recognize properly drafted limitations on those warranties.

Implied warranties have received a great deal of attention in cases dealing with computer systems. The Uniform Commercial Code (UCC) implied warranty of fitness for the intended purpose has been deemed most applicable.

However, since the UCC applies only to the sale of goods, the first question which must be addressed is whether the thing supplied constitutes "goods" under the UCC.

Little question exists that computer hardware constitutes goods under the UCC. Likewise, programming and systems design clearly constitute services, and are not goods. If it is found that what was supplied constitutes goods, then one or both of the UCC implied warranties may apply. UCC section 2-315 provides for an implied warranty of fitness for the intended purpose. UCC section 2-314 implies a warranty of merchantability "if the seller is a merchant with respect to goods of that kind." The defendant may claim that the plaintiff has waived the implied warranties by failing to reject the equipment. However, there is no requirement that the purchaser immediately reject the computer system, so long as the rejection is made within a reasonable amount of time. The purchaser is not considered to have accepted the equipmment merely by installing it. A "reasonable amount of time" includes the opportunity to determine if it meets the requirements of the contract.

Extent of liability under the agreement

Though a vendor may be liable for breach of an express warranty or for breach of am implied warranty, such liability may be limited by the terms of the agreement. UCC section 2-316 allows implied warranties to be disclaimed, as long as the disclaimer is in writing, conspicuous, and specifically mentions the merchantability if it seeks to limit the implied warranty. However, the provision limiting damages must be clearly and specifically drawn. Damages may also be limited by setting a maximum dollar amount of recovery (usually no more than the amount paid by the purchaser on the contract.)

Negligence

Complaints against computer vendors frequently contain a claim of negligence. It should be recognized that the presence of a valid contract may not only affect a claim of negligence, but may, in fact, preclude it. One who performs an act under a contractual agreement in a negligent manner is liable to the contracting party for damages and not under a theory of negligence.

COPYRIGHTS

As the computer market booms the ever increasing need for software is being filled by many enterprising individuals who formulate thoughts and ideas into reality on a computer system.

Unfortunately, the yearning of software purchasers for smarter, faster, and more 'user friendly' programs may turn into a crying need because of the void in the copyright laws as they affect software. This discussion on copyrights draws no clear conclusions due to the fact that the judicial system in the United States sways from one end of the spectrum, that of protecting the author, to the other, allowing free use of any and all software that can be begged, borrowed, or stolen. The word stolen is used intentionally because the unauthorized use of anything can and does constitute theft.

The intent behind the copyright law of the American Constitution recognized a need to protect authors' control of original written expression, and gave Congress the power to grant authors a limited monopoly in their works. The law of copyright is designed to advance social welfare by maximizing the public availability of literature, music, and the arts. Because a regime under which all creative works were in the public domain would discourage creative effort, copyright seeks to guarantee some economic return to the copyright owner by granting the owner a limited monopoly. Copyright, theoretically, thus achieves its goal of maximizing availability by balancing the desire of the public for uninhibited access to intellectual works against the desire of the creators of those works for financial reward.

The original concept of the copyright law was geared toward the publishing industry with no provision made for second order technology. The fair use doctrine, initially developed in the court system and enacted in the Copyright Revision Act of 1976 was created to restrict the control that a copyright gives an author over the publication, distribution, and performance of copyrighted work (for purposes of scholarship, criticism, etc.) The doctrine emerged as judges perceived that vesting excessive control of a work in the copyright owner would often inhibit the creative efforts of a second author who would otherwise have relied upon the copyrighted work. Thus, by tiliting the balance toward public access and away from private gain, the fair use doctrine allows authors to borrow reasonably from copyrighted works to create new expressions. At the heart of the fair use doctrine lies one major shortcoming -- an inability to define precisely the extent to which one may borrow from a copyrighted work. The courts have never applied the fair use doctine with any predictability, even before the emergence of second-order technology.

The area of copyrighting software lies in a vast wasteland at the moment. New hardware technology is covered by the patent laws. Software is an "intangible" and does not

easily fit under the copyright or patent laws. There have been many recent instances of software copyright cases emerge through the judicial system. Judges presiding over these cases are setting precedents on each and every case because they have no concrete guidelines on which to establish their rulings. The following paragraphs examine some recent decisions in the area of copyrighting software.

The Patent and Trademark Office granted Merrill Lynch Pierce, Fenner & Smith, Inc. Patent No. 4,346,442 for a data processing system involved in its Cash Management Account. The lengthy patent describes in great detail a series of transactions in combination to produce the desired result. This patent was granted based on the decision by the Supreme Court in Diamond v. Diehr, 450 U.S. 175 (1981), where the court held that a patent claim could not be denied "simply because it uses a mathematical formula, computer program or digital computer" and that a new combination of steps in a process could be patented, even though the steps were well known and commonly used before the combination was discovered. The Merrill Lynch patent appears to be based on a number of simple steps, using simple mathematical formulas combined to produce results that in many cases could be achieved by hand or with the aid of a fast calculator. Merrill Lynch told other firms that it will require them to obtain licenses for accounting systems that are substantially similar or else face court action. Whether such lawsuits would be successful is debatable since a valid patent cannot be issued where the invention is "obvious to one skilled in the art." After the issuance of the patent Paine Webber, Inc. filed a law suit in the federal district court in Delaware seeking to invalidate Merrill Lynch's patent.

Another case to set precedent was the case of GCA Corp. v. Chance, No. C-82-1063 in the Northern District of California. This case issued an preliminary injunction against three former employees of GCA Corp. preventing them from using diagnostic and operating system programs belonging to GCA. In the judge's decision it was decided that the registration of a source code protects the object code and that limited distribution of an object code is not publication under the 1976 Copyright Act. In accordance with an informal ruling of the Copyright Office that source code, but not object code, can receive a clean copyright registration. GCA filed the source code prior to bringing this action. The filing consisted of the first 25 and last 25 pages of the programs involved which was considered sufficient for registration but did not disclose the inner workings of the programs.

The judge held that source code "falls with the protection of copyright laws as a work of authorship fixed in any tangible medium of expression from which it can be perceived, reproduced or other wise communicated" and, because the object code is "decryption" of the copyrighted source code, the two are to be treated as one work. Therefore, the copyright of the source code protects the object code as well.

Among the most vexing problems for software developers and marketers is determining what legal procedures are available to protect proprietary rights. In light of the decisions made by the courts in recent cases it would be wise to look at the rules published by the Copyright office. The Patent and Trademark Office has stated guidelines in their Manual Of Patent Examining Procedures' New Section 2110. The "Rule of Doubt" allows registration of computer programs when there is no law of the subject. This rule was not a law enacted by Congress or devised by the courts. In essence it stipulates that when the Copyright office isn't sure whether a work is registerable or not, it will accept it for registration, but will advise the author that it is in doubt. When programs were first registered, they were usually filed in source code. However, unless there are special circumstances, everything registered with the Copyright Office is available for visual examination (but not copying) by anyone so there has been an increasing tendency to supply object code with the registration application. In the procedure outlining submission of programs for copyright consideration it requires that source code be submitted for review. Noting that copyright examiners are not programmers and that it is extremely difficult to examine programs in other that source code form (such as object code which is all zeros and one) to decide whether the program contains copyrightable authorship, the Examining Division has concluded that "the best representation" is a printout in source code format. In cases where the author is unable or unwilling to deposit the printout in source form, the Examining Division will register the program under the Rule of Doubt "upon receipt of a letter from the applicant assuring that the work as deposited contains copyrightable authorship." This letter is essentially a statement by the proprietor that the work is original and not a copy of someone else's program. There is some talk of establishing a special unit to handle computer programs.

RIGHTS OF BUSINESSES IN REGARDS TO IN HOUSE DEVELOPED SOFTWARE

Where an employee is hired with special skills, such as a programmer or analyst noncompetition, disclosure, and ownership clauses are often a part of the employment agreement. Most states enforce this kind of agreement, although some, California being the most notable exception, do not.

Where they are enforceable, the courts require that they be reasonable as to length of time and to the geographic area in which the employee is prohibited from competing. When the courts find that the agreement is unreasonable decisions can go two ways. Some say that if the agreement is unreasonable the entire clause in unenforceable. The better view, however, is that the court cuts the restrictions back to what it finds reasonable.

The drafting of such clauses is difficult at best, and the decision on whether to attempt their enforcement is also tough. The more important the employee is to the company's operations, the more likely a clause of this nature will be enforced. Good legal counsel is essential.

Programmers are the most likely candidates for the nondisclosure and ownership clauses in employment agreements. Because programmers are often non typical workers in regards to their hours it is imperative that an employment agreement stipulate total ownership of any product developed during the term of employment. A programmer uses his creativity and intellect in performing his or her job and in cases of a non-friendly leave taking, they can use this argument to support partial ownership of the product they wrote while on the payroll of a company.

In many cases the employee may object to signing such an agreement because of "moonlighting" situations that they are involved in. A carefully drafted employment agreement will allow for such situations while still protecting the company's assets. In the case where a conflict of interest policy exists, this is a mute point.

SUMMARY

The computer industry continues to grow at an incredible rate and hence our legal problems and questions grow. Until the legislative processes accelerate to handle the industry's unique situations, all members of the professions are counted upon to police their own corners of the world. Congress is only in the infantile stages of studying the many facets of computer law and cannot move quickly enough to avoid the flood forcing its way through the judicial system.

In these uncertain times their is no substitute for sound legal advice on the issues before us. As economic uncertainties affect us, the legal uncertainties affect our installations every day. Contracts should be reviewed by competant legal counsel, company assets (such as software) should be protected by all means possible, and employees should be encouraged to leave company secrets behind.

If you wish for further information on topics covered in this paper such as case histories, please feel free to contact me.

IMAGE STRATEGY - WHERE ARE WE GOING?

NANCY COLWELL

HEWLETT-PACKARD

In March of 1982, an IMAGE survey was distributed to the attendees of the San Antonio IUG conference. What has happened to it? Has Hewlett-Packard used it or thrown it away? The survey, thanks to all of you, has given Hewlett-Packard a better understanding of how our customers are using the product and the problem areas of the current IMAGE. This talk will address the current IMAGE strategy as it exists today. The talk will go as follows:

I. San Antonio Survey Results.

   a. Discussion on the top 5 highest rated problems within each section of the survey

II. Solutions to current problems.

   a. Enhancements to IMAGE

   b. Solutions to current IMAGE problems

III. Discussion of the IMAGE strategy that is in place today.

IV. Questions and answers.

DICTIONARY/3000 - An Inside View

Nancy Colwell
Hewlett-Packard

This paper discusses what passive key links are within Dictionary/3000, and how they are used. By understanding passive key links, you will understand two things: Why Hewlett-Packard only supports certain utilities when used against the Data Dictionary, and why IND has increased the functionality of the DICTDBM program to include the "UTIL" function.

A passive key link is a link that is not controlled by the IMAGE software. However, this link is just as important as the links controlled by IMAGE. Passive links exist because of nature of the Dictionary/3000 structure requirements. These requirements will become more clear as we discuss the different types of passive links and how Dictionary/3000 uses them.

There are three types of passive key links maintained in the data dictionary. They are as follows:

1. Description key link
2. Path key link
3. Sort key link

DESCRIPTION KEY:

The passive key link called description key maintains all of the descriptions within the data dictionary and links the descriptions with their respective owners. The owner may by an entity, an association, or a relationship. As you already know, a description may be included with the response to any CREATE, ADD, or RELATE prompt. This information however, is not manditory. The person running DICTDBM may hit a <CARRIAGE RETURN> in response to the DESCRIPTION prompt. If this is the case, no description will be entered. Because of this unique process, the IND lab chose not to use the IMAGE path mechinism. If the IMAGE path was chosen to link the description text with its respective owner, there would be a tremendous waist of space in the dictionary for every entity, association, or relationship that did not require a description to be linked with it. The lab chose instead to maintain the link inside the Dictionary/3000 software.

A control data set is used to house the description keys in addition to the path and sort keys. This data set houses "current key value". When a description link is needed, the Dictionary/3000 software retrieves "current key value" from this control data set and places this value in the appropriate sets in order to create a complete link (e.g. an entity name with it's

respective description). This linkage is used in commands such as CREATE, ADD, RELEATE, DISPLAY, MODIFY, CHANGE, and UPDATE.

PATH KEY:

A passive key link called path key maintains all of the IMAGE and KSAM path information within Dictionary/3000. When needed, the dictionary software will retrieve this value and places it into the necessary data sets to provide path information. When elements are associated to a KSAM file or an IMAGE data set, DICTDBM will prompt for "KEY ELEMENT" and "PATH MASTER FILE". This information is necessary to create the path key link. The path key link is used by DICTDBC to create an IMAGE data base and assign the appropriate path structure between the master and detail data sets. The SHOW command in DICTDBM uses this information to provide you with the data base path information in the output dispaly. Also, path key information is used with HP Inform groups to set up their links.

Current key value for the path key is also maintained in the control data set. When needed, the Dictionary/3000 software will retrieve this value and place it into the necessary data sets to provide path information.

SORT KEY:

A passive key link called sort key maintains all of the IMAGE sort informtion within the data dictionary. This information is used when a sort link for IMAGE must be maintained within the data dictionary A sort key link is set up when adding elements to a file that is of type "DETL", DICTDBM will prompt for "PATH SORT ELEMENT". The sort key link is used when the SHOW command is requested on a file of type "DETL" or "BASE". The sort key link is also used when creating an IMAGE data base using DICTDBC. Again, current value is kept in the control data set.

As you can see, passive key links are very important to the data dictionary structure. Altering, erasing, or duplicating the key values will corrupt your data dictionary. Because of this potential problem, IND chose to limit the utilities supported by HP as shown in the "WARNING" on pahe ii of the Dictionary/3000 Reference Manual. This warning lists all utilities that are safe to use with your data dictionary. You should not use any utility that is not included in that list, the integrity of your data dictionary could be destroyed.

The "UTIL" enhancement to DICTDBM was provided to allow you to clean up passive key links that are no longer needed. When you you PURGE an ENTITY, DELETE an ASSOCIATION, or REMOVE a RELATIONSHIP, the information is deleted but the passive link

value is not reused. This will build up superflous passive key link entries within your data dictionary. DICTDBM,UTIL will unload your data dictionary and reassign new passive key link values as it reloads the information back into the data dictionary. DICTDBM,UTIL is used to "clean-up" your data dictionary and should be run on occasion to maintain a clean dictionary.

Quality Assurance: Keys to Higher Performance and Profit

Jane Copeland
Tymlabs Inc.

In today's fast moving society we often find ourselves placed in a situation where we have to produce systems in very short periods of time. In order to meet these deadlines we usually spend less time on standards, design, and documentation. In a one-time case this method doesn't generally do any lasting damage, but too often quick and dirty systems produced to meet unrealistic demands caused by bad managerial planning, are left indefinitely in production environments. This can cause a multitude of different problems. I doubt that the additional and sometimes continuing costs that result from this type of situation have ever been accurately assessed.

There are several fundamental areas which should be addressed in every DP installation to avoid this type of situation. Efficient systems design and utilization of resources can only be accomplished if some consideration is given to the following fundamentals, before embarking on a major project:

a) Capacity Planning

Two basic items determine the point when a computer system reaches its limitation: the attributes of the computer system, and the quality and performance of the applications software. Efficient systems design guarantees maximum utilization of the hardware. Unfortunately, the continuing trend of lower priced hardware has prompted many companies to purchase more hardware when approaching capacity without regard to optimization of the systems they already own. Adequate capacity planning eliminates the waste caused by reactionary measures that are often taken to provide necessary support resources.

b) Programmer Productivity

Education plays an important role in insuring that maximum advantage is taken of system capabilities and tools to provide the highest degree of productivity. Applications designed and implemented without sufficient knowledge of the computer system and the application itself generally result in solutions that require constant modifications, thereby reducing productivity. An equally detrimental effect of those applications is the negative impact they have on overall system performance.

c) Standards and Procedures

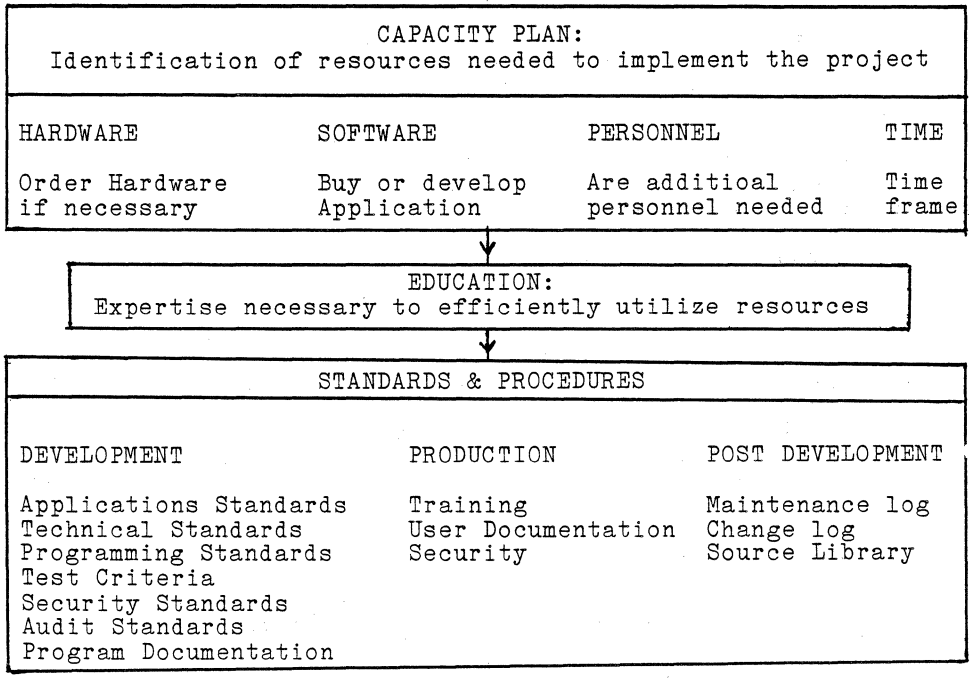
Formal standards and procedures provide the regimentation necessary to ensure continuity throughout the MIS environment. This continuity is the foundation for

implementation and maintenance of quality solutions. Constant monitering is required to enforce adherence to the company's procedures standards. Companies operating without formalized standards and procedures greatly impair their overall efficiency. When an employee leaves, a great deal of information and experience is lost. if comprehensive standards and procedures are in place, this loss is minimized and the training time required for a replacement employee is greatly reduced. The benefits are also realized in the areas of daily operations and development projects.

Currently most peoples perception of Quality Assurance is limited to software development and hinges on the implementation of standards and procedures. In my opinion this is a great mistake, and one that should be rectified immediately. Successful Quality Assurance in the MIS environment can only be achieved by placing equal importance upon Capacity Planning, Education, and the implementation of Standards and Procedures.

The following is a graphic presentation of a Quality Assurance Plan which contains all three items and shows their interelationship:

QUALITY ASSURANCE PLAN



Although it is evident that standards and procedures provide the main basis for software quality control, we can see that ot is of little use without the basic functions of planning and education taking place beforehand.

A great many DP managers realize the necessity of these activities, but have a difficult time convincing their management of the cost benefits. These benefits are far reaching and it would be difficult to assign a specific cost savings. Here is a list of areas where cost can be minimized as a result of comrehensive Planning and Quality Assurance:

Less implementation time
Reliability
Maintainability
Less training time for new employees
Maximization of resources
Audit capabilities/prevention of fraud
Data integrity
Ease of use
Less run time
Less personnel
Less production problems and bugs
Greater Security

Since we rely on computers to provide us with accurate and timely information that is utilized in making management decisions, it is important that we make every attempt to insure the reliability and security of this function. Incorrect or late information can be the difference between a business surviving of failing in todays economy. Can we afford to defer implementation of a Quality Assurance program any longer, with the excuse it costs too much?

Programming & Productivity Tools as Pathways to User Satisfaction

J. Roger Daugherty

Vice President
Sun Software, Inc.

THE CHALLENGE

One of the greatest challenges facing data processing managers today is the ever-growing population of sophisticated users, computer products, and applications, which so often exceed the limited budgets and resources of the formal computer organizations in our companies. This is especially true in HP3000 shops which have not shared traditionally in the larger budgets of main-frame data processing facilities.

Growing demand for service is not the only problem with user satisfaction; it is being compounded daily by a rapidly increasing level of user awareness, technical training, and computer expertise. It is, in fact, progressively more difficult to find fellow company employees who don't possess at least a passing familiarity with the power and potential utility of computers in the workplace. These same users who may have been extremely difficult to sell originally on the use of computers, can become surprising vocal about the failure of computer systems to address new and often more complex problems in the office.

Finally, there is the proliferation of computers and related products. For little more than the cost of a new typewriter, individual department heads are offered computer power today comparable to a fairly large computer installation just a few years ago. Obviously this threat to the data processing management of a company goes beyond a mere challenge. It goes to the heart of the leadership role of data processing management to determine the direction, style, and inter-relationships of all data processing applications and facilities within the company. The company itself is exposed to many of the same potential problems which have lead to the adoption of strong data processing organizations: data storage redundancy; data discrepancies; hardware incompatiabilities; and loss of management control.

The real problem was well described by Dan McCracken last fall in a keynote address to Data Training '82. "We have the kinds of problems in applications development that many people would like to have in their areas," said McCracken. "There's a lot more work out there than we know how to do with conventional methods." In his address he went on to praise the recent developments in data base methods and inquiry systems, particularly higher level languages and other programmer productivity efforts.

THE SOLUTION

Fortunately, there exists throughout the Hewlett Packard 3000 community a wide array of tools and systems to help data processing management meet the above challenge. These are variously sold, given away, or bundled in with the hardware, and range from the more simplistic system operations assistance programs to the most sophisticated whole systems generators. Some are appropriate for use by operators and system managers, most by programmers and analysts, and a few by actual users.

The problem for a data processing manager, therefore, is to become aware of and remain familiar with all the available tools and products, and to select (and cost justify) the tools to be utilized in an individual HP3000 facility. Two important aspects of this selection process are the appropriateness and the acceptability of each tool by the intended users. The following commentary provides a framework for both the justification and acceptance of these tools in user organizations, and it can be used by individual managers in considering various products.

The other purpose of this discussion is to aid the manager in the actual selection process, first by reviewing most of the available tools and products along with their essential functions and applications levels. In addition the final section suggests some specific evaluation standards which the manager may use in assessing the tools for their potential utility.

This discussion is limited to two classes of tools and aides, those which help data processing users to do their regular work more effectively by improving system performance or access, and tools which change the development process by actually performing for the user many of their regular work tasks. The latter category ranges from simple code generation tools and automatic documentors to almost complete system generators. These products also vary in their potential for non-data processing personnel to utilize them.

	Aides/Tools	System Generator Aides
Data Processing Personnel	Operation Aides System Utilities Data Base Utilities Interactive Editors, De-bug Aids	Code Generators General Report Writers Transaction Processors Documentors
Other Users	Data Dictionaries	High Level Report Writers System Generators

In the following discussion there is no attempt made to identify either the sources or vendors for each of the products except to note some of the HP bundled or sold products in each category for the sake of comparisons. Similarly only a few items of particular importance from the User Contributed Library are included, since it is assumed that most HP system users who have access to this material are already in possession of a full description of that Library.

Individual product costs, fees, and maintenance schedules are not described, either, along with the vendors since such information is regularly available in user literature and at meetings and conventions of user groups. Also, it is not the purpose of this material to compare or contrast these individual products, but rather to familiarize the reader with all the possiblilities and to suggest ways and situations where they may help all of us to be more effective managers.

THE TOOLS

OPERATIONS AIDES - In addition to the regular store of tools provided by HP with each installation, the User Contributed Library is full of aides of this type. Examples are WIZ, SLEEPER, and many others designed to make system operation easier, faster, and better organized. Many of these tools can contribute to significant savings in operator resources and greatly enhanced system performance. Other tools and aides available are:

MPEX
WORKFLOW
MORPHEUS/3000
DISPATCH 3000

SYSTEM UTILITIES - Again both HP and the User Contributed Library have made substantial contributions to this area, designed to optimize system performance and optimally manage available system resources. One problem which must be noted here is the historical frailty of contributed Systems Utilities which frequently encroach on HP system methods and proprietary procedures. Changes to the HP operating system or other support systems are often, therefore, not compatible with the systems in the Contributed Library, since no one is accountably responsible for supporting the tool. Other system oriented utilities available are:

SECURITY 3000
LOOK/3000
RAS/3000

DATA BASE UTILITIES - IMAGE- Most readers are no doubt familiar with the popular history of IMAGE, and are aware that it is well supported by HP. There have also been several support and utility aides in the Contributed Library in the past. In addition, there exists an impressive group of utilities which generally go well beyond the tools normally available with IMAGE. They are:

ADAGER
DBPLUS
DBAUDIT
DBUTIL(HP)
QSKETCH
SUPRTOOL

INTERACTIVE EDITORS, DE-BUG AIDES- Programmer Productivity, often measured in such crude terms as "lines of code per unit of time", was probably first addressed with these tools. Most were originally intended to circumvent the lost productivity in conventional HP programming caused by moving among the regular editor, the compilers, the segmenter and test jobs. Generally they have become much more than simple work savers and they enjoy considerable popularity among HP system users:

EDITOR (HP)
EDIT+
QEDIT
TESS/AIDE
SPEEDEDIT
VTEST

DATA DICTIONARIES - IMAGE - Data Dictionaries serve two vital purposes. First they act as a common data definition base for programmers and users alike to better understand and document the data bases of an organization. This may have been their first use, but most are now also deeply integrated into the Transaction Processors and System Generators described below. The most common of these which are generally associated with IMAGE data bases are:

Dictionary3000 (HP)
Dictionary PLUS
EZD
QSCHEMA

CODE GENERATORS - Another method of increasing programmer productivity, and therefore, hopefully, user satisfaction, has been to develop systems which produce program source code based on application descriptions, data definitions and report requirements. Subsequent code modification may or may not be required. These products differ from the Transaction Processors described below by the rather arbitrary distinction that they are not PRIMARILY intended to interface between users and data bases which have been previously defined, rather they are intended to stand alone. Some of these are:

AZ7
COBBLER
COBGEN (Contributed)
PROTOS
SCSS

GENERAL REPORT WRITERS As with much HP Utility software, most of the Report Writers available seem to be primarily intended to interface with IMAGE data bases, although KSAM, ISAM, and other file interfaces are frequently available. General report writers are distinguished from higher level ones by the fact that they require the user to have a fair degree of technical skill in order to use the tool. The best known of these are:

QUERY (HP)
QUIZ
REPORT/3000
REX

TRANSACTIONS PROCESSORS - As noted above most of these are specifically oriented to act as an alternative to standard source code programming of applications systems to update and support IMAGE (and other) files. HP has given particular attention to this in the development of their RAPID System. The most well known such tools are:

CP 3000
DE 3000
QUICK
TRANSACT 3000

DOCUMENTORS - Many of the tools and products described in this disucssion can legitimately claim to greatly improve documentation levels and subsequently enchance overall user satisfaction. Indeed this is one of their most important justifications as productivity tools. There are a few systems, however, which exist solely for the purpose of creating documentation, quickly and cheaply:

BAD 3000
CAD 3000
The DOCUMENTOR

HIGHER LEVEL REPORT WRITERS - These tools are often sufficiently sophisticated that the user may not require much technical expertise in order to be able to utilize them. They typically rely heavily on the use of pre-defined data dictionaries (see above), are likely menu-driven, and can often take the place of whole report programs and systems at a reasonably complex level. Some of these are:

CRW
INFORM 3000 (HP)
PAL 3000

SYSTEM GENERATORS - As was already stated, System Generators are intended to replace the entire computer programming process, and are often designed to be utilized by the actual end user. They are usually large, somewhat more expensive than report writers and transaction processors, and are generally very good at allowing subsequent modification. These tools address directly the primary key to user satisfaction—completing a good system in the least amount of time. Some of them are:

GENASYS 3000
INSIGHT II
PRIDE
RAPID (HP)

JUSTIFICATION FOR THE TOOLS

Obviously each purchase/utilization decision must be made by the individual data processing manager in the light of current applications demands and existing job loads. Four areas of investigation are suggested, however, for the manager seeking to explain to upper level personnel why the product or system is required.

First, as has been discussed repeatedly above, is the need to improve programmer productivity. Statistics abound concerning the cost of programmers in an organization today, the cost of replacement, and the number of unfilled jobs. It is generally not too difficult to demonstrate that the cost of most tools is easily replaced by a few man-months of programmer time saved. Also, more productive programmers are also more content, and more likely to remain in their positions, and more able to attract other productive programmers.

Enhanced system performance is certainly a goal of productivity tools, and an easy cost justification for the acquisition of such products. Of course, the productivity tool may have operating overhead implications, but many feel that the opportunity to more directly access users' specific requirements more than makes up for whatever performance impact such tools may have.

Various studies have determined that programmers often spend only 20-30% of their time doing actual analysis and programming. Many tools are specifically intended to increase this figure by minimizing the time programmers spend on non-essential tasks such as documentation, testing, and user interfacing, while seeing that these functions are still performed in an adequate manner. Some tools go one step further, to remove the programming requirement entirely from the technician's responsibility, allowing them to concentrate on design considerations such as data storage, reports, and overall user requirements.

Often the most important argument for the use of these products is the intended improvement of the utility and applicability of user systems to actual information and system requirements. Ease of modification, better documentation, improved system access, and easier communications with data processing "experts" are all sited as the improved systems aspects these tools allow.

USER ACCEPTANCE

A key to user satisfaction with productivity and system generation tools lies in their acceptance by the users. As with any new or different way of doing business, users may resist new approaches until it has been demonstrated to them that the new tools will benefit them in ways which matter to them. The following arguments have been partially presented above, particularly with regard to the system generation products. Users need to be shown they are true.

1. The use of the tools will result in better overall system performance and response since the precise needs of the user can be better addressed. In other words, less time will be used performing tasks or producing reports not needed or desired by the current user.

2. Similarly use of the tools will result in better systems, reports, and user interfaces. Access will be better, simpler and the new systems developed will reflect current needs and problems better than the old.

3. Applications systems can be delivered, tested, accepted just that much more quickly than without these tools!

4. With most of the tools, finished documentation is created simultaneously with the completion of testing and implementation. This documentation is normally in standard format and often ready for review by senior managment and outside auditors.. These are very powerful arguments in favor of these tools and their acceptance by the users.

5. If the user is paying for the actual development cost, or the subsequent usage costs of the computer system, it should be relatively easy to convince him of the cost savings that better, more efficient development and systems will involve.

6. Applications generated using these tools are more easily modified subsequent to installation. This includes not just reduced labor, but also the improved communications among the original developers, the users, and subsequent maintenance and modifications personnel.

EVALUATION OF THE PRODUCTS

As indicated above, there has been no attempt made herein to judge or otherwise compare individual tools and products. Indeed every shop and application is likely to have very different needs and preferences with respect to these tools. Therefore, the following is a list of comparitive measures which individuals may wish to make in evaluating any of the above tools. Even with these, the relative importance of each parameter will vary with each site and application.

1. Will productivity actually be increased for all programmers and analysts? That is, is the product usable, complete, and relevant for the personnel and applications involved?

2. Can significant cost savings be demonstrated? In the long term? In the short term? How about hidden costs in training, conversions, and programmer unhappiness?

3. Does the tool or product contribute to the establishement of systems standards for the organization?

4. Does the product really ease future maintenance costs and difficulty? Or is it likely to be more of a problem to maintain itself than the applications it is being acquired to support?

5. Does the product support or produce finished, acceptable documentation? If so, is it equally easily modified and corrected?

6. Will the resulting applications systems produced or supported really be of better quality? See 1-5 above.

7. Does the system or product, or its use, increase the overall business knowledge of the person doing the development or the end user? A good product will do this as well as all the above.

SUMMARY

All of the tools and products mentioned in the above discussion can and have met the needs of organizations and users in many different environments and circumstances. Their utility in each organization in dependent on all of the factors which have been described. What is important to remember is that these tools can have significant impact in any organization if they are properly selected and used within each company.

In fact, these products have the potential for considerable cost savings in many organizations, and through their use the data processing manager has a very real opportunity to improve the total data processing user satisfaction level throughout any organization. Well chosen tools will serve the entire user organization well.

BIBLIOGRAPHY

1. Gesink, Don. "Improving Programmer Productivity - An Integrated Approach" in Proceeding of the IUG, 1982.

2. Kahlow, Ron and Loughin, Curt. HP3000 Productivity Tools, A Comparison of. Data Systems Inc., 1982.

3. Zientara, Marguerite. "Keynoter Calls DPers 'Victims of Own Success' " in ComputerWorld, November 15, 1982.

4. Kull, David. "Generators Multiply Output" in Computer Decisions, January 1983.

5. Black, David R. COBOL Program Generator

6. Parks, Merle. "Productivity Tools Enable Users to Obtain Better (not more) Code" in Software News, February 1983.

7. Bryce, Milton. "MISMANAGEMENT" in InfoSystems, February 1983.

DESIGN AND PROGRAM OPTIMIZATION TO SUPPORT OVER 300 TERMINALS

N. M. DEMOS
DEMOS COMPUTER SYSTEMS, INCORPORATED

I. INTRODUCTION

This paper examines various techniques from the simple to some very sophisticated to maintain excellent response time and throughput in a very heavily loaded system. Simple single terminal programs will be presented first, followed by a complex example required to achieve performance objectives. These techniques would also apply on the Series 64 when the user wished to employ more than 128 terminals. The 128 terminal limitation is imposed because of maximum table sizes in MPE IV. It will be demonstrated that many more terminals can be supported if certain conditions are met. These conditions include:

1. A large number of short, relatively simple, transactions.

2. Application disc Input/Output time and CPU process time are adequate to support the load.

3. A large number of terminals support the same or a limited set of applications.

General system design guidelines will be presented, followed by an overview of an example system as it was developed. The following areas will be described:

1. Process handling, particularly to overlap Image I/O.

2. Using extra data segments as scratch files to save disc I/O.

3. Inter Process Communications Files and no-wait I/O.

4. Communications considerations.

5. Handling many terminals per process.

6. Special programming techniques for efficiency and to multi-thread terminals.

It is possible within certain constraints to handle a large number of terminals on the HP3000. The terminals must be dedicated.

The application must not consume an undue amount of computer resource per terminal. An efficient monitor program must be employed to manage the system resources effectively. Application code must be written to interface to the monitor. In addition, as the number of terminals increases, a way must be found to reduce the overhead implicit in Hewlett Packard's present terminal handling hardware and software. While these constraints may be violated in minor ways, it must be remembered that the HP3000 has a finite computing and file access capability.

Applications requiring a high degree of computing and a large number of file accesses for each entry from a terminal are not good candidates for the approach presented here.

In given applications, specialized hardware may have to be interfaced to the HP3000. It is not the purpose of this presentation to emphasize or advocate a particular hardware approach, although one hardware solution will be presented in the context of the software approach described.

It is the objective of this paper to present management and programming techniques that lead to improved performance on the 3000. While it was necessary to employ almost all techniques to implement the case presented, most of them have a more general applicability. Some of them lead to better, more maintainable programs in all cases.

II. PLANNING FOR EFFICIENCY

A. Languages.

Sometimes the application and its performance requirement dictate the programming language to be used. SPL was employed in the example presented later because it was the only reasonable choice in the environment that existed when the application was implemented. Today, COBOL II might be a reasonable choice for parts of the application, particularly on a Series 64. In most cases, any one of a number of languages will do the job well. It is important that the programming manager establish standard languages and standards for each language for a commercial installation. He should pick one standard language, e.g. COBOL II and possibly a secondary language for exceptions; this secondary language is usually SPL but we have seen FORTRAN used effectively. Then the programming manager should establish usage standards for each language that lead to structured and efficient code. If these standards are designed and implemented well, the resulting programs will be easily debugged, readily maintainable and reasonably efficient. Efficiency thus becomes a no cost by-product of good programming techniques. In most cases, efficient code

is as easy or easier to write as inefficient code, if only the programmer knows what the correct techniques are.

Standards and guidelines also apply to other areas, such as blocking factors for disc files, when to use direct versus serial access, when to use other file constructs such as KSAM and/or IMAGE. It is beyond the scope of this paper to state just what the standards should be, although some will be suggested by the material presented later. The language manuals have some suggestions. Also refer to the bibliography at the end of this paper.

B. IMAGE - Data base design.
IMAGE is an excellent, relatively easy to use tool for storing, maintaining and retrieving data. However, it is easy to make design and programming errors using Image that result in very poor performance. Disc I/O time is expressed in milliseconds, instruction time in microseconds. Remember this relationship (1000 to 1) when considering where to spend time optimizing. A heavily used data base should be simple in design yet responsive to the way it is used. These two requirements are often in conflict. Adding new records and deleting records are relatively slow with Image. If data is volatile in this way, it may be necessary to use a non-Image file structure or delay adding or deleting records until the system is less heavily used. This can be easily done for deleting - a delete flag in the record can be implemented and all processing programs can check for this flag and treat the record as though it was not there. Adding records presents a harder problem. Records can be written to a regular MPE file and the adding done later if it is not important to have these records on the data base in "real" time. If the system controls the assigning of the search item value, then adequate records can be pre-built during slack times so that a much more efficient update can be done at the time the record data is available. Sorted chains, particularly long ones present a severe problem when adding or deleting records. It may be necessary to perform a programmatic sort when accessing the records to avoid sorted chains.

The single most important approach to Image efficiency is good data base design. Unfortunately this is not easy and may not even be feasible until some actual experience is obtained from actual use of the application system. Most programmers underestimate the number of data base intrinsic calls that are made during a production run by a factor of 8 or 10. Because data base calls are often CPU intensive as well as requiring disc access, performance may be poor. It is

beyond the scope of this paper to explore further the methodology of good data base design, but there are good documents available on this subject (see the bibliography).

C. Hardware.
It is often beyond the capability of the systems analyst or programmer to effect the hardware resource available for his application. However, he should be aware of the characteristics of the disc drives on the system he uses. On the Series III, Look Ahead Seeks should be enabled if he has more than one drive. To achieve the same effect on the other (HP-IB) machines, the system has to employ the newer drives or have more than one master drive.

The distribution of the Image data sets among the disc drives may be important and can be controlled. Good distribution can often result in significantly better performance at a small cost.

III. GUIDELINES

There are some activities that consume a large amount of system resource, particularly CPU time. These activities are certainly necessary, but often their use can be reduced. Below are some of the activities to analyze for performance improvement with the most resource intensive presented first.

A. Sign-ons.
A sign-on may take as many as 1000 disc I/O's. Where possible, users should sign-on and stay a long time. Accounts, groups and users should be assigned with this in mind. Avoid automatic BYE's with UDC's when the user will probably sign right back on to the same user and account. There are other ways to reduce sign-ons (refer to dedicated terminals below).

B. RUN's (and CREATE's).
These are also heavy users of disc I/O and the CPU. A file has to be accessed and a data stack built and assigned in virtual storage.

C. File and Data Base Opens.
These require a directory search and buffer assignment.

D. Disc I/O.
A disc I/O almost always takes at least 15 milliseconds and the average is over double that figure. This compares to the typical machine instruction that completes in a matter of a few microseconds.

In most cases where performance improvement is desired, disc I/O's particularly Image I/O's, present a more cost effective area for improvement.

E. Terminal I/O.
Most terminal I/O (except through the new ATP or the INP) is expensive in CPU time because of the character interrupts it generates in the system. Each character input or output requires a CPU interrupt and code execution to place it in a buffer (or extract it for output). The overhead expense is compounded for V/3000 screens. Remember also that each new V/3000 screen must come from disc.

F. Intrinsic calls.
Most intrinsic calls are generalized and do some parameter checking. While we are not suggesting that they not be used, sometimes several calls can be combined into one call.

G. Loop constraints.
Program loops particularly when parsing terminal input or searching tables, can consume an inordinate amount of CPU time.

IV. SPECIAL TECHNIQUES

A. Dedicated terminals.
In some cases, where a set of terminals is to be used for dedicated applications or a limited subset of applications, it may be desirable to treat the terminals as programmatically controlled devices rather than have users sign-on. For example, where a user typically dials in, completes a short transaction and hangs up, there may be as much as 10 times the system resource used in signing on, RUNning the program and opening files as in processing the transaction. It is possible to CREATE a process that opens the required files and re-initializes itself for each new user. There are other situations where dedicated terminals may be effective. It is one way to breach the 128 terminal limitation. This limit is imposed because each session (or job) requires at least two processes, one for the Command Interpreter and one for the program itself. The first can be eliminated if a STREAM command is used to CREATE several processes, one for each dedicated terminal.

B. Process Handling.
Process Handling is very effective in situations where certain functions involving I/O can be performed in parallel. It is also useful where a resource, e.g. data base, needs to be controlled in special ways, for

example, holding off adds (DBPUT's) during periods of heavy activity. Jim May's article "Programming for Performance" in the July/December 1982 of the Journal of the HP-3000 IUG gives several good examples of this approach. Because no-wait I/O cannot be done against an Image data base, process handling may be particularly appropriate, in this case, to attain better response time. The new Inter Process Communication file structure makes this type of implementation easier. Because a process that reads an IPC file will wait until there is data to be read, no activates or explicit use of data segments is required. Also, if the number of I/O buffers assigned to the file equals or exceeds the number of blocks of data in the IPC file at any one time, no disc I/O will occur.

C. SPL Routines.
SPL routines are effective ways of accomplishing functions not available in other languages and/or standardizing certain often repeated functions.

If they are loaded into an SL file (possibly the system SL file, although this has its drawbacks) standardization and modification of certain functions can be achieved without having to recompile the programs.

D. Internal Sorts.
There are various ways of sorting data that avoid a stand-alone disc to disc sort, which is quite inefficient because of the disc I/O's required. For a trivial number of records (for example, the records in a short Image chain), the program could bring them all into memory and search for the first, etc. Various methods of using the sort efficiently by accessing it programmatically are discussed in "Programming for Performance" previously mentioned.

E. Delayed or Anticipatory Data Base Adds and Deletes.
As mentioned previously, add and deletes of Image records are time consuming. Where performance is critical, it is often necessary to avoid adding and deleting records from the same process that is handling terminal I/O. Flagging records for later deletion is relatively easy. For records to be added, a separate process (see process handling above) can be used, or if the application allows, the records can be pre-allocated or written to an MPE file for adding during slack time. Remember that if a separate process is employed, at the very minimum, the dataset will be locked during the add. Process handling in this case improves response time but is no help in reducing system load.

F. Extra Data Segment as Intermediate Storage.
In many cases, large tables or short files are required to be accessed again and again. It is often advantageous to pre-load these into one or more extra data segments and access them from there during program execution. If the memory is available, they will remain in memory, which is much more efficient than accessing them from disc.

G. Negative DB and Dynamic Area in the Stack.
In SPL, it is possible to dynamically obtain additional memory if the requirement for a large table varies widely from program execution to execution. With the DLSIZE intrinsic memory can be made available as needed and in reasonable increments. Sometimes, two large dynamic storage areas are required. In this case, it is possible to use the ZSIZE intrinsic to get more stack space and then move the Q and S registers upward to free the room required. Remember that there is a MAXDATA limitation.

H. FREADDIR and Number of Buffers.
Sometimes, when only a limited number of records are required and used from an MPE file, over and over again, it is possible to set the number of buffers high enough so that only a few reads are required. Most of the time, the data is already available in the buffer. Unfortunately, there was a bug introduced with MPE IV that made the file system use only one buffer when accessing a file with FREADDIR's (fixed in the Q-MIT). This method is particularly attractive when it is known that FREADDIR's are employed and it is impractical to change the source code.

I. Blocking.
An analysis of the frequency of use of certain files and their more common method of access might lead to the implementation of a different block size. This applies not only to MPE files, but also to Image data sets. Remember that the schema parameter BLOCKMAX can be used to set blocking individually for each dataset if appropriate.

J. Disc Allocation.
When fine tuning the system for performance, it may be desirable not to include the systems disc in the class DISC. Files that are seldom accessed can be RESTORE'd to this drive by using the DEV= option. Virtual memory can be distributed among several or all drives to help balance disc accesses. Then files that are heavily accessed (particularly Image data sets) can be assigned to different drives (as above) to minimize head contention.

K. Program Segmentation.
This is often discussed as a means of making memory management more efficient. As systems have larger and larger memories, this becomes less important. On a small memory machine, if a large heavily used program is divided into 4K segments and good locality maintained, it will probably improve efficiency.

L. Data Conversion.
It helps to maintain numeric data on disc in the format most often required by programs. Data heavily used for calculations should not be stored in ASCII. It will have to be converted every time it is accessed. In the typical 3000, a significant percent of CPU cycles are devoted to data conversion.

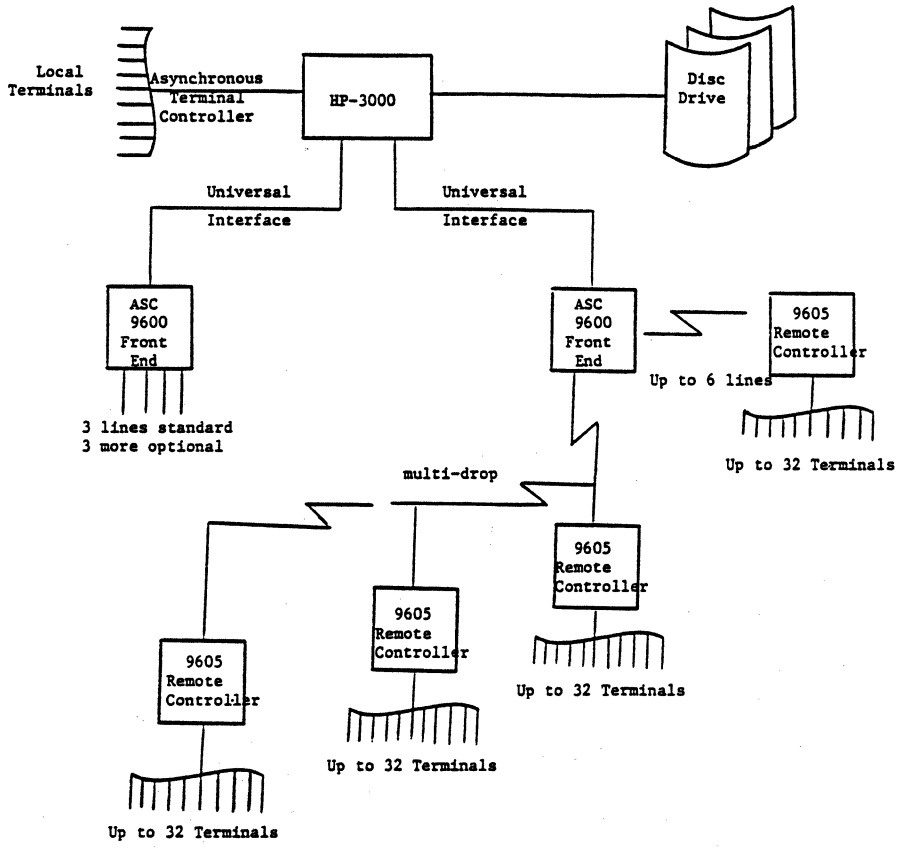
V. A CASE STUDY - HISTORY

In mid 1976, an opportunity arose to design and implement a multi-store Point of Sale System on the HP3000. It was decided that the best implementation would employ a CRT, printer and cash drawer combination that would look to the system like any other asynchronous terminal.

The terminal was connected to an intelligent, remote controller which acted as a control and buffering device so that data could be transmitted synchronously over leased telephone lines. An SDLC-like protocol was implemented for the communications network so that maximum efficiency could be achieved. Multi-drop capability was included so that a number of remote controllers could share one line. At the host site, a "front end" communications controller was designed to support the remote controllers. In order to interface to the HP3000 hardware and operating system, the front end was made to look like the HP programmable controller. This was the product offered at the time by Hewlett Packard for communication between a 3000 and the HP1000 line of computers. We were therefore able to use the Universal Interface Driver (IOREM0) supplied as part of the MPE operating system. The Universal Interface (the same as the line printer controller) on the HP3000 is a parallel interface. It supports programmed versus direct I/O so that CPU interference is much less than with the asychronous terminal controller, which employs direct I/O (each character requires an interrupt).

We now had all the hardware and system software components required to make the system operate. Next we had to design and implement a software monitor.

DCS COMMUNICATION SYSTEM
HARDWARE FEATURES
SERIES III



NOTES: ASC 9600 is the Front End Controller for remote terminals, supporting up to six lines. Each line will support up to fifteen remote controllers.

VI. IMPLEMENTATION OF THE SOFTWARE MONITOR

When we designed the software monitor to achieve our goal of supporting a large number of terminals on the 3000, we kept in mind several objectives. These were:

A. We wished to maintain the integrity of MPE.

B. Coding should not be too difficult and the programs should be easily maintained.

C. Response time at the terminal should be in the one to three second range; it should not exceed six seconds.

D. Memory utilization should be minimized.

E. As local terminals were to be attached to the system through the Asynchronous Terminal Controller, the monitor must relinquish control when it is not performing useful activity.

The HP3000 with its MPE operating system imposed an environment that included the following characteristics:

A. Although I/O no-wait was implemented, there was no way to determine if an I/O operation was complete and do any processing if it was not.

B. There was no way for the monitor (which was the father process) to suspend and be awakened by EITHER I/O completion or an activation by a son process (meaning that its task was complete and it had data for the monitor).

In this context, several decisions were made, as follows:

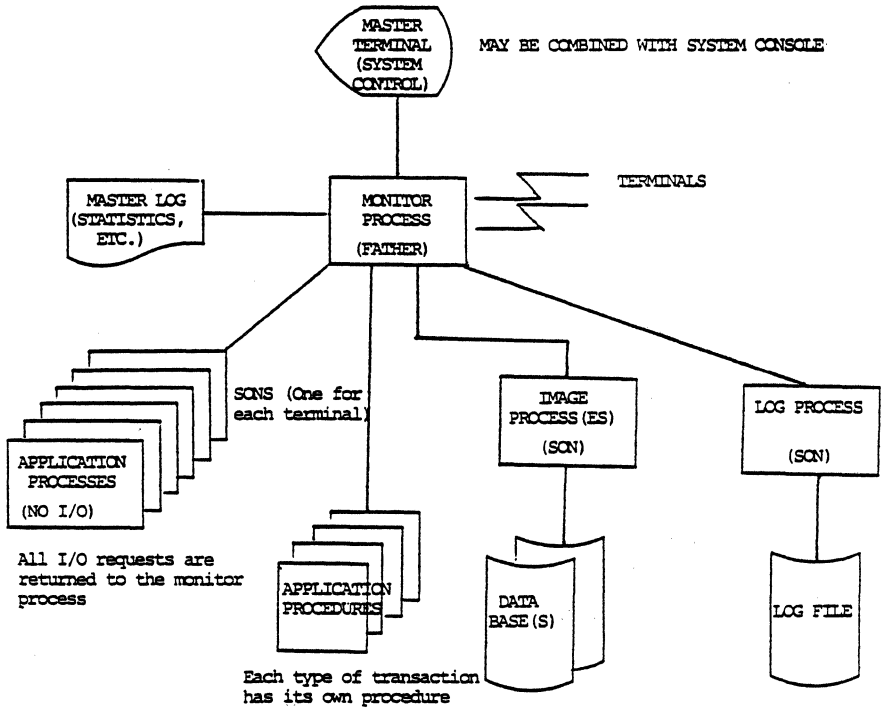
A. Because of the required efficiency, the monitor and the application code would be programmed in SPL.

B. The monitor would handle all input and output to the communications front end, but handle no other I/O directly (except communication with the monitor terminal that would be used to control the system).

C. Each terminal would be assigned its own process; this process would do no input-output of its own.

D. The Image Data Base Management System would be used for those files where random access was required.

HP3000 - USING A MONITOR FOR EFFICIENCY



E. There would be separate, asynchronously running processes for Image access and other disc access (the latter required mostly for a Transaction Logging file).

F. All interprocess communication would be through extra data Segments.

G. All transfers of control would have to be explicitly handled through the father process.

H. Queues would be set up and maintained for the Image and Disc Access processes so that the monitor could perform other activities while an individual terminal might be waiting for Image or Disc Access.

With these decisions in place, a main process loop was designed. That started with a read to the front end and ended with an IOWAIT if no activity was present in the subsystem. The monitor runs in the BS queue so that, at this point (IOWAIT), other sessions could perform activity. Also, whenever there was activity and a process was started, the monitor would suspend when no data was available from the front end.

Approximately six months after this system was designed, the hardware and software were brought up for final test purposes. With some minor adjustments (mostly involved in debugging firmware in the front end and controllers), the system worked satisfactorily. About six weeks later, it was put into production.

The original design goal was to support eighty terminals. The next requirement was to support 100 - 120 terminals. This was accomplished through implementation of an additional front end to lessen the load on the first front end, and some changes in the monitor program. The major change was the implementation of read and write queues (buffers) for the front end I/O. This was necessary, not only to smooth momentary peak loads, but also because the front ends had limited buffer space. The implementation of write buffers allowed processing to continue even though there was a momentary peak load on the output side of the system. These changes, an increase in memory size, and other minor fine tuning adjustments allowed us to increase capacity as required at that time.

The next requirement was to support up to 250 terminals doing the same applications. At this point we realized that a process for each terminal was no longer effective. First, each process requires a MINIMUM overhead of 2048 bytes, for its stack, not considering working storage for data movement and calculations. Secondly, research indicated that even without paging to and from disk, it required an absolute minimum of eight milliseconds for a monitor to activate a process, suspend and then to have the son process reactivate the father. This figure holds with no

processing being done in the son. For these two reasons, it was decided to discontinue the process per terminal approach and use a procedure of the monitor for application code. A place to store the data for each terminal and a method of determining the location in the code where processing for any given terminal was to continue were required. In other words, some of the housekeeping that MPE performed automatically through process handling had to be simulated by other means.

Because Q-relative storage could be used for data required between input-output requests, we only needed a special place to hold data during I/O requests. This was accomplished by assigning 70-word blocks in negative DB. These were managed by the monitor. Because the application required that additional, considerably more dynamic, memory be allocated to store print lines for a given transaction, a method was devised to manipulate the original Q location upward and use the available space as temporary storage. This was also handled by the monitor, although individual procedures could request data and release data in this area directly. In a later version, we used extra data segments for this type of storage. This is a more flexible approach, although not quite as efficient because DMOVIN's and DMOVOUT's are required.

The other requirement was handled by calling a special exit subroutine before leaving the procedure. This special subroutine stored its own exit address in the special data area mentioned above. In complementary fashion, a start subroutine, always called at the beginning of the procedure, retrieved this exit address and used it as its own exit address. Some additional housekeeping was required, but this is essentially how the procedure operated. This required a coding discipline where no input/output could be done in a procedure or subroutine called by this application procedure. For obvious reasons, it was also desirable to keep the data required during input/output operations to an absolute minimum. In general, these were the only programming constraints necessary to implement this approach.

Because it would have taken considerable recoding to make the disc handling process a procedure and because access to it was not required often enough to have a significant impact on performance, it was left as a separate process. Because there was no way of doing no-wait I/O with the Image files, it was necessary to leave the Image code as a separate, independently running process. The above changes were implemented and results were as expected. Thrashing was eliminated and response time was improved with 160 terminals active. In this application environment and the way this system operates, it is felt that 200 terminals operating at once would be feasible.

250 terminals active at one time is probably the peak for this application on a Series III. The application has grown more complex over the years. Also a communication link has been

established with an IBM system for credit card verification. At this point, the application is process bound on the Series III during peak periods. The following might improve performance:

A. Restructure the Image Data Base.

B. Take random processing out of Image and use a custom random access subsystem.

C. Review and restructure the main process loop in the monitor to make sure that the most likely event is tested for first and that tests are made by groups. Using the new IPC file feature will make the process loop simpler and will save some CPU time. At the same time, we would review and, if possible, refine our methods for terminal address calculations, etc.

On the Series III, we realize we are pushing the limits of the HP3000 and we will be looking for ways to offload some of the processing. One way involves linking one or more HP3000's together via a DS-link and splitting the loads between them. A Series 64 will also allow us to radically increase the number of terminals we can support.

We would like to note that we have a system with over 1000 terminals configured. However, the activity per terminal is very low.

Although the system presented here may be more exotic than some users can foresee implementing, it has fulfilled its purpose and has performed with a larger throughput than anyone had originally envisioned. System uptime has been more than satisfactory, and processing has been very inexpensive in its environment. We feel confident in recommending that any situation requiring a large number of dedicated terminals where the application processing is not too extensive, be programmed using the above approach. The key criterion is the requirement for a large number of dedicated terminals where each input from a CRT terminal does not require a significant amount of computational or disc processing. Under these conditions an excellent response time will be achieved, with economic hardware and software costs.

BIBLIOGRAPHY

1. Programming for Performance by Jim May. July/December 1983 (VOL 5, No. 3, 4) of the Journal of the HP3000 International Users Group.

2. MPE Internals for Neophytes by Robert M. Green. July/August 1982 (VOL 2, Issue 4) of Interact HP3000 International Users Group, Incorporated.

3. Overview of Optimizing (On-line and Batch) by Robert M. Green. Proceedings of the 1982 International Meeting HP3000 International Users Group.

4. System Performance and Optimization Techniques for the HP3000 by John E. Hulme. Proceedings of the 1982 International Meeting HP3000 International Users Group.

5. Image/Cobol. Practical Guidelines, by David J. Greer. Proceedings of the 1982 International Meeting HP3000 International Users Group.

6. Pitfalls of Testing Disc I/O Performance by HP Anonymous. Proceedings of the Seldom Met Users Group. Robelle Consulting, Ltd.

SYNCHRONOUS COMMUNICATION ON THE HP 3000

N. M. DEMOS
DEMOS COMPUTER SYSTEMS, INCORPORATED

The interface between a terminal and a computer has remained a mystery to many. This is only as it should be. The typical user should not have to know how data gets to and from his terminal. It is sufficient that it does. Others, such as those responsible for configuring and ordering hardware, need only a very basic knowledge. Sometimes a more thorough knowledge permits the implementation of applications not otherwise possible. We will try to enlighten our reader so that he will have enough background to be able to know how to proceed further if he is so inclined.

Sometimes an apparently simple requirement cannot be easily accomplished with standard HP software packages. We found this to be true with the communications (DSN) products. We were given a requirement last year to interface an application to an IBM system for purposes of credit card verification. The transaction had to be handled interactively over an IBM Bisynchronous link operating in contention mode. None of the HP products were suitable. The closest was DSN/Interactive Mainframe Facility. However it required that we appear to the IBM host as an IBM 3270. This may have been possible, but IMF has a high CPU overhead. We decided to use the CS/3000 subsystem directly. The CS/3000 subsystem is not an HP product as such. It can be used if the appropriate HP communication subsystem is purchased. After an introduction describing the HP communications software products, we will further describe CS/3000.

Most terminals are attached to the HP 3000 through a local ("hardwire") or remote (over telephone line) link using RS-232C protocol. R232C is a specification detailing how the electrial signals will be generated. The terminals operate asynchronously, that is, each character is individually timed. There are two inherent problems with this kind of interface. There is no automatic checking and retransmission if the data is not received correctly. Secondly, there is a speed limit of 120 characters per second for this method when it is used over telephone lines. This is not an absolute limit as new technologies become available, but because each character must be individually synchronized, there is an inherent speed constraint.

In order to attain increased speeds the synchronous method of transmission was implemented. This method relies on the devices at both ends to stay synchronized for an entire message (typically a line of data, but often much more). With synchronous transmission very high speeds are possible.

For example, the HP 3000 supports speeds up to 7000 characters per second in synchronous transmission. As speeds increased, it became more important to automatically check for transmission errors and re-transmit when an error was detected. Because of this requirement and the need to maintain synchronization over an entire message, the hardware must be more sophisticated and is therefore more expensive than hardware used only for asynchronous transmission, but the payoff is in the higher speeds attainable.

On the HP 3000, the INP (Interface Network Processor) is the current interface for synchronous communications. On the Series III there is also the SSLC (Synchronous Single Line Controller) and the HSI. The former is not as flexible as the INP (speeds only to 1200 characters per second for example) and requires more process time (or "overhead") on the 3000. The HSI (High Speed Interface) is used only for a Series III talking to another in the same building. The INP has its own microprocessor and buffers to reduce overhead on the processor. It converts data to and from one synchronous line.

On the 2624 and 2626 terminals, there is a special feature that allows these terminals to operate synchronously. The 2645 also has a board that gives the same capability. HP has a new product (the Multipoint Cluster Controller - product number 2333A) that allows any standard asynchronous terminal (up to 16 per controller) to take advantage of synchronous transmission. Therefore to use synchronous transmission the hardware items required are terminals as described above and at least one INP on the 3000. To transmit over telephone lines, the appropriate modems (or digital units) and lines are required.

The most common and one of the oldest methods ("protocols") used to take advantage of synchronous communications is the IBM Binary Synchronous Communications Protocol and its variations. HP uses this protocol for most of its communications products, both because it is well known and because HP wishes to support communications with IBM main frames. HP also supports a newer protocol patterned after IBM's Synchronous Data Link Communications (SDLC) protocol as well as SDLC itself.

In order to implement various synchronous communication capabilities on the 3000, HP developed a subsystem called CS/3000. This consists of a series of intrinsics analagous to the file system intrinsics (FOPEN, FREAD, FWRITE, FCONTROL, etc.) that can be used to define the desired communications protocol, logical device, speed, etc. (COPEN) and receive and send data (CREAD and CWRITE). These intrinsics are defined in Section I of the Communications Handbook (Part number 30000-90105) although little information is given about their use. HP has an internal document that gives much more information on how to use these

Intrinsics (CS/3000, External Reference Specifications). There is no HP product available called CS/3000. Instead the user obtains CS/3000 by purchasing any HP communications product. In most cases the user can employ one of the standard products to accomplish his goals.

Multipoint Terminal Software (DSN/MTS) is the subsystem required to support HP synchronous terminals on the 3000. Except for some limited models of IBM's 3270, it only supports HP terminals. However, standard asynchronous terminals are supported through the Multipoint Cluster Controller mentioned above. MTS also supports hardwired terminals using HP's Data Link method (mostly for the 307X family of factory data collection terminals). Its major advantage for the non-factory user is its capability to support remote clusters of terminals with a single high speed line. It uses Binary Synchronous protocol (multipoint) except in special cases. It does support an asynchronous protocol for local application (but you still need a special product for your terminal to use it this way). It is also a method to operate terminals at 600 characters or 1200 characters per second (speeds not otherwise available) on the Series III if that is imperative.

The Remote Job Entry (DSN/RJE) subsystem is used to make the 3000 appear to an IBM mainframe like an IBM 2780 or 3780 remote job entry work station. It can also be used to communicate between two 3000's but DSN/DS (see below) is usually a better choice for this purpose. Data may be sent from and received on almost all 3000 peripherals. The RJE concept involves batch processing, so this subsystem cannot be used for interactive communication.

RJE uses IBM Binary Synchronous protocol in what is called "Contention" mode. This means that each end of the line (the 3000 or the IBM mainframe) bids for the line. There is no polling as in MTS. Therefore only one 3000 can be on the line at a time communicating to one IBM mainframe. However, dial-up as well as leased telephone lines are supported so that connections to various host at different times can be made if required.

Multileaving Remote Job Entry (DSN/MRJE) is a more sophisticated version of RJE. MRJE emulates work stations that work with the IBM HASP, HASPII, ASP, JES2, and JES3 Job Entry Systems. This package should be used in place of RJE when it is available on the host system and multiple users on the 3000 wish to submit jobs for processing. Because the 3000 emulates an intelligent system (computer) much more flexibility is available to control job submission, control and routing. It also cannot support interactive communication with an IBM mainframe because of the nature of the IBM system it emulates. MRJE uses an option of IBM Binary Synchronous protocol called "Conversational" mode.

This mode makes the communication link more efficient in cases when there is data to be sent both ways. The normal receiving station can send data along with its protocol acknowledgements (this is where the term "Multileaving" comes from).

The Interactive Mainframe Facility (DSN/IMF) is the only other HP product that communicates with an IBM host. As its name implies, it is used to communicate interactively. It makes the 3000 look like an IBM 3270 type cluster controller. A terminal on the 3000 can look like an IBM 3270 type terminal (Pass Thru Mode) or a 3000 program can access the host with special intrinsics (Program Access Mode). Program Access Mode is a powerful tool. It is the only HP supported method to programatically communicate with an IBM mainframe interactively. IMF uses either Binary Synchronous or IBM SDLC to communicate with the IBM host. In both cases it supports multi-point, that is, the 3000 can be one of many stations on the same line.

Distributed Systems (DSN/DS) is the most powerful of the HP communication products. It is used for transferrring data between two 3000's and between a 3000 and another HP computer (1000, 250, etc.). It also supports compatible terminals on Public Data Networks (Telenet and Tymnet). With DS a user on one HP computer can log onto another HP computer, use any peripheral on the other computer, transfer files to and from the other computer, access a data base on the other computer and write programs that talk to each other directly over the communications link. The link may be either local ("hardwired") or over telephone equipment (remote). Remote access may be either over a leased line or by dial-up. Programs do not have to be changed to use most of the capabilities of DS. Extensions of the FILE Command and additional commands, e.g. REMOTE, provide these capabilities in a user friendly fashion. DS uses the Binary Synchronous protocol except for Public Data Network access, in which case the X.25 standard protocol is employed.

Originally all these communications products were to be programmed using the CS/3000 subsystem. Unfortunately, performance considerations forced HP to compromise. Both DS and IMF use modules that do not employ CS/3000 for interface to the INP.

What does the user do if none of the HP communications systems meets his needs? First of all he should make certain that he cannot use one of these systems, either by asking for changes at the host or by redesigning his application. For example, if he requires interactive access to an IBM mainframe, his first option is to ask that he be given access as a 3270. He can then implement his application through IMF. If this does not apply and he is a knowledgeable user he may be able to use the CS/3000 intrinsics. This was done successfully in one case where it was required to communicate interactively with an IBM mainframe using the Binary Synchronous protocol in contention mode.

The application required an interactive response for credit card verification and data capture. IBM Binary Synchronous protocol using contention mode was the communications mode specified. This is the same method and protocol used by RJE, but RJE assumes communication to the host in a batch mode (remember RJE stands for Remote Job Entry). Although data can be entered through $STDIN, the host looks upon the input as a job submission. Also RJE could not be integrated into the rest of the application on the 3000, it is meant to be a stand-alone program. It may have been possible to have the IBM host communicate with the 3000 as a 3270 cluster controller, with the 3000 using the Interactive Mainframe Facility (DSN/IMF). IMF has intrinsics that can be used to make a program look like a 3270 terminal. However, IMF is an elaborate subsystem with a high CPU overhead. The IMF Program Access Mode (the way it accesses the host programatically) requires the programmer to use and format a dummy terminal screen. This requires considerable programming as well as adding to the overhead.

Performance was an important consideration because of the load on the system. Only because there was no feasible alternative was it decided to use CS/3000. This approach required considerable knowledge of communications and experimentation with some parameters of the CS intrinsics. It was successful in achieving its applications goals.

In order for the programmer to use CS/3000 he must first configure his INP (SSLC or HSI if he wishes to use one of those interfaces on a Series III). The configuration is fairly straight forward, and if he does make a mistake, most of the configuration parameters can be overriden with the CLINE command. Because we were going to use IBM Binary Synchronous protocol (Bisynch) we followed the specifications for RJE in the Systems Manager/Systems Supervisors manual. Our communications equipment consisted of an INP, its modem cable, a modem (201C equivalent) and a leased line. Therefore the TYPE in the configuration was 17 (INP) and the SUBTYPE was 1 (Synchronous, nonswitched line with modem). Switched means dial up, non-switched mmeans leased line. Common sense applied to the rest of the parameters and they can be overriden by the CLINE command and/or the COPEN intrinsic anyway. There is only one driver for the INP, it is called IOINP0. This is only the driver skeleton. The rest of the protocol is contained in a download file. For IBM Bisynch contention mode this is CSDBSC0. This is the default. It comes with DSN/DS and DSN/RJE (and maybe other DSN products). If you are trying to use CS/3000 with other versions of Bisynch or with SDLC, beware. You will have to match your protocol and mode with one of the DSN products and purchase it to get the correct download file. A friendly, communications trained HP SE is invaluable at this point.

If you have any questions about the required communications equipment, you should refer to the HP Guidebook to Data Communications and/or the HP Communications Handbook (see bibliography). The Guidebook has valuable information on how communications equipment works and its various varieties. The Handbook has more specific information relating to HP products (be sure you have the 4/81 or later version). If you have all your equipment and HP software in place, you can now begin to write your application. It will help if you code the entire application using a programatically controlled terminal to simulate the communications link. In that way you would know that the rest of the code works before stepping into the unknown. You will have to use SPL or an SPL subprogram to establish the communications interface.

Most CS/3000 intrinsics have direct equivalents in the file system intrinsics (see figure one). Treat the communications line like another file except use the CS intrinsics instead. There are several differences that you must take into account. The COPEN has parameters describing the communications environment that are quite different from the parameters for FOPEN although some of the parameters are similar. The COPEN intrinsic format is shown on page 1-30 of the Communications Handbook. Either the formal designator or the device parameter is required. The rest are optional. The three options fields and some other parameters are explained on pages 1-13 and 1-29.

IBM Bisynch requires that each series of messages be terminated by an EOT. When writing to the line (CWRITE) you must end with an EOT before you finish or can read (CREAD) data. This is done with a CCONTROL (linenumber, 1, param). When reading, a CCG condition means that an EOT has been recieved instead of data. You must keep posting CREAD's until you get the CCG condition.

Error handling needs to be more elaborate than normally used for the file system. The 200-250 series of error numbers are particularly important because they may indicate problems on the line and/or problems at the other end of the line. In particular you may see many 205, 207, and 217 errors. 205 occurs during CWRITE and means that the other end is trying to send data. A CREAD should immediately be done to receive this data. A 207 typically ocurs during a CWRITE if the line takes several "hits" or the computer at the other end goes down. A 217 error is the result of a CREAD failure when the other end goes down. Most other errors in the 200-250 series can be corrected by repeating the operation. Other errors are hardware or software prolems at the local site. They are classified as to probable cause on page 1-21 of the Communications handbook.

CS/3000 INTRINSICS AND COMMANDS

Intrinsic (Command)	Similar File Intrinsic	Comments
COPEN	FOPEN	Returns line number similar to file number
CCLOSE	FCLOSE	
CREAD	FREAD	
CWRITE	FWRITE	
CGETINFO	FGETINFO	
CGROUPINFO	None	Supplies information about a remote group on a multipoint line
CCHECK	FCHECK	
PRINT'LINE'INFO	PRINT'FILE'INFO	
CCONTROL	FCONTROL	
CPOLLIST	None	Creates or updates a poll list for multipoint lines
IODONTWAIT	IODONTWAIT	Intrinsic is the same and has the same purpose (complete I/O)
IOWAIT	IOWAIT	Intrinsic is the same and has the same purpose (complete I/O)
CLINE (Command)	FILE	
CRESET (Command)	RESET	
SHOWCOM (Command)	None	Shows line errors since last COPEN

FIGURE 1

If a large number of errors in the 200-250 series occur and the link and modems seem to be working correctly, one or more entries in this MISCARRAY parameters of the COPEN should be analyzed and possibly changed. Type 5, number of error recovery retries, may be increased from its default of 6 if it is known that the line may be poor and the application can afford the increased time required.

There are five different timeouts that can be changed. The receive timeout occurs only for a CREAD when the remote does not send text, an EOT or a TTD (temporary text delay) after the first text message is received. The default of 20 seconds seems more than adequate. It could be shortened under most conditions. It is set at too short a time, you will get many 209 errors on the CREAD's. The local timeout has nothing to do with the remote or the line. If you will be waiting for data from the remote for long periods of time it should be disabled. It is used to prevent one session or job from monopolizing the line. The timeout occurs when no CS intrinsic is activated during the time period. The default is 60 seconds. If it occurs (error 155) it will disconnect the line. Connect time is the time that the CS systems will wait for the modem to indicate that it is ready (DSR or Data Set Ready Line comes on). It's default is 900 seconds. It should be set to about 15 seconds in situations where it is known that the modem and line are operational. If it occurs (error 151) it will disconnect the line. If the line is disconnected it should be CCLOSE'd and COPEN'ed to continue.

Response timeout is the amount of time the system will wait for a response to a local action during a CWRITE. Its default is 2 seconds for a primary contention station and 3 seconds for a secondary contention station. An error 207, driver retry exhausted, will be received if this timeout occurs. The line bid timeout, (error 217) occurs durng CREADS. It means that the remote has no data to send. The default is 60 seconds. The timeout may or may anot indicate a true error, depending on whether data is expected or not. All these timeouts are described on pages D-12 and D-13 of the Communications Handbook (the line bid timeout is there called the "Wait timeout"). Note that they are described as they apply to RJE, but the description is the best we have seen in HP documentation.

The checking of identification sequences is part of the Bisynch protocol. This checking may be inhibited be setting bit one of the COPTIONS field on. Both local and remote ID sequences go in the array IDLIST or the configured default is used. ID sequences are checked as soon as the COPEN becomes effective, i.e. connection is made. Another important part of the COPTIONS field is the "local mode" subfield, settings 0-4 refer to various options using Bisynch or SDLC, setting 5 and 6 refer to HPDLC. If this subfield is not compatible with other COPEN parameters, particularly the protocol subfields of the AOPTIONS parameters, an open failure will result.

The protocol subfield of the AOPTIONS parameter must be compatible with other options of the COPEN and the download file (or driver if the INP is not being used). If contention mode Bisynch (protocol) is desired the download file is CSDBSC0, which is the default. In other cases you will have to be careful to match the download file to the protocol. This may require some investigation. All download files start with CSD, are 515 words long and reside in PUB.SYS.

Like the file system, the normal mode of operation has the CS/3000 intrinsic not return control to the program until the I/O is complete. By setting bit 15 in AOPTIONS parameter to 1 the program will gain control before the I/O operation completes and the program must issue an IOWAIT or IODONTWAIT instrinsic to complete the operation. There are several differences in the way this operates in CS/3000. There are no special IOWAIT and IODONTWAIT instructions for CS/3000. Because the line numbers returned by COPEN and the file numbers returned by FOPEN are mutually exclusive, an IOWAIT with a first parameter of 0 can be used to wait on the first I/O that completes, file or line. Secondly, several I/O requests (up to a maximum of 14, 7 reads and/or 7 writes on the INP) to the same line number can be made without issuing intervening IOWAIT's.

The NUMBUFFERS parameters of the COPEN is used to set the number of I/O's that may be queued (negative value) or the number that may be queued and buffered (positive value). If you use buffers, you do not have to do the COPEN's in privileged mode to do concurrent (no-wait) I/O. Having several CREAD's outstanding for a line number may be convenient (remember that the EOT requires one), but it doesn't seem appropriate when doing CWRITE's because of the difficulty taking action for the appropriate record on errors in transmission. It could be done if the sequence of records transmitted were not important and the data was kept available for re-transmission. Each CREAD and CWRITE must eventually be paired with an IOWAIT or IODONTWAIT that shows completion unless I/O is aborted by a CCONTROL with 0 as the first parameter. If the CCONTROL returns an error code of 64, an IOWAIT must be issued because the I/O has already completed.

There are several aids that can be used to help diagnose problems. For the INP the program DSM.PUB.SYS can be run to test the HP hardware, the download files and, to a limited extent, the associated communications equipment. Its operation is described in the INP Diagnostic Procedures Manual. Part No. 30010-90002.

There is also a CSTRACE facility that can be invoked by bit 2 of the COPTIONS parameter or the CLINE command. It writes to a file describing in detail every action taken by the

communications subsystem. These records may then be printed by CSDUMP.PUB.SYS. There are several options that can be used. We suggest that the first attempt to employ CSTRACE be with no options set. CSTRACE generates a large number of entries for every intrinsic called, even if no options are used.

If the programmer still has difficulty after using the diagnostic tools mentioned above, he might try to use a line monitor to see what is actually happening on the line. He may be able to borrow one from HP or rent one from a supplier.

The HP communications products work well, but sometimes it is necessary or advantageous to use the CS/3000 instrinsics directly. This is possible but it requires considerable technical expertise and even more patience. Communications technology itself is complicated. The user is advised to use one of the HP products if at all feasible. Only if he has the necessary resources should he proceed with direct use of CS/3000.

HP has a number of communication subsystems for the 3000. The user should carefully evaluate his requirements before he chooses one because most of them are not too flexible in their implementation. However, if the user employs them as HP intended, he will find that they usually perform in a more than adequate fashion.

Bibliography

Binary Synchronous Protocol, IBM form no GA27-3004-02

Communications Handbook, HP part no. 30000-90105

CS/3000, HP External Reference Specifications

Guidebook to Data Communications, HP part no. 5955-1715

HP3000 Data Communications Products - Specification Guide
HP part no. 5953-7444

HP 30010A/30020A Intelligent Network Process (INP)
Diagnostic Procedures Manual HP part no. 30010-90002

HOW MUCH SECURITY IS ENOUGH?

A Method for Determining Appropriate Levels of Protection at Mini-Computer Sites

Doug DeVries
Hewlett-Packard
Corporate Data Center

Abstract

This session will deal with computer security, addressing the issue of adequate protection of mini-computer assets. The focus will be on physical security, so that site managers can know how many resources to invest to prevent (and recover from) potential disasters.

Mini-computer sites have varying security needs. Many managers need a method to quickly evaluate risks and decide on adequate protection. Also, they need a method that will get beyond subjective opinions and stand up to auditors' scrutiny.

This workshop will present a process that has been applied at over a dozen HP sites. It uses a structured Delphi technique, involving a diverse group of people and a functional analysis of the data center. The evaluation process can be completed with minimum manpower in 1-2 elapsed weeks, and can also increase DP support and security awareness in the user organization.

Outline

I. Overview
   * Introduction and Definitions
   * Underlying Reasons for Security
II. The Risk Analysis Process
   * Identify assets
   * Evaluate risks
   * Develop alternatives for reducing risks
   * Make recommendations, and implement
III. Forms

Introduction

Security is a topic that creates ambivalent feelings. On the one hand, there is a sense of obligation to do at least something -- to appease the auditors, to reduce the vague anxieties of upper

managers, or to avoid the catastrophies periodically described in ComputerWorld. From this perspective there is a strong element of external "shoulds" to create "security".

On the other hand, "getting security" is not all that easy. When you sit down and seriously look at computer security, most of the textbook answers are "overkill" for mini-computer sites. Also, security tends to be an overhead item that does not directly help the productivity of the computer center, and it normally seems like a large task that can be delayed until "tomorrow". Also, the probability of "bad events" actually happening seems so low that it is hardly worth the effort to deal with them.

So, what is security? Why is it important? And how can you figure out what you need to do -- at an appropriate cost? This paper will try to suggest some answers to these questions.

A Definition

In its simplist form, the purpose of security is:

TO INSURE CONTINUITY AND INTEGRITY OF DATA-PROCESSING ASSETS,

THROUGH AN APPROPRIATE LEVEL OF PROTECTIVE MEASURES.

By "continuity" we mean reliable, continuous processing, with minimimum disruptions due to extended downtime. By "integrity" we mean that virtually all the computerized information is correct, complete, and private.

Underlying Reasons for Security

The concern over computer security has multiplied in the past dozen years, primarily because the role of data-processing has changed.

In the old days, when a small percentage of an organization's information was on computers, there was accordingly a low level of vulnerability. But as the number of applications has goes up -- and the majority of business functions have become computerized -- the vulnerability has gone up too. Thus, computer security is essentially a "hitch-hiker" -- the increased need for protection reflects the changing role of computers in business and government.

Computer use implies computer dependence. Your user population assumes that your operation will have "continuity" and "integrity" -- and management up-the-line will be angry if these assumptions are not met.

Thus, computer security is important not merely because of the auditors or some other external pressure. Instead, security is ingrained in the nature of the service you're providing. Security boils down to one key area of "good management" of computer resources.

As a DP manager, then, you'll be trying to determine what's an "appropriate" level of protection. Looking at security seriously means that you'll be weighing multiple factors, such as:

EDP ROLE IN THIS ORGANIZATION

MINIMAL ---------- EXTENSIVE

PROBABILITY OF "BAD THINGS" OCCURING

LOW ---------- HIGH

HUMAN IMPACT OF SECURITY

CONVENIENCE ---------- PROTECTION

COSTS TO IMPLEMENT

MINOR ---------- MAJOR

The methodology listed below provides a system for "crystallizing" security-related information. It falls under the title of "Risk Analysis". The outputs of the risk analysis should give the responsible managers enough information to make informed decisions about risks and cost-benefit trade-offs. The ultimate goal of all this security effort is to get an "optimum" amount of protection at a reasonable cost.

RISK ANALYSIS

** A Process for Determining Appropriate Levels of Protection **

1) GETTING STARTED

There are several things to do, as you get ready to embark:

** Review this text, examine the forms, and make sure that you understand the sequence of activities. Then lay out a general schedule for your work, and decide who you want to be involved.

** Meet with your boss(es) to review your work plan. The goal here is to make your management aware of the risk analysis activity. You want them on your side from the beginning, since they will be involved in the final decisions on accepting risks and/or allocating resources to reduce the vulnerability.

** Talk with people you would like to involve, in order to get their agreement and time commitment.

2) IDENTIFY ASSETS

Using attachment RA-1 as a starting point, identify the value and extent of the data-processing function in your division.

This process is like taking an inventory. However, do not spend an undue amount of time to get detailed money/number figures. Ranges and approximates are enough.

This activity will probably involve the building (facilities) engineers, as well as DP people who are responsible for hardware, contracts, and applications interface. Getting estimates for the time required to re-build/re-install can have an impact both on this risk-analysis process and on your broader disaster-recovery plans.

After collecting the asset data, prepare the information in a neat format and make enough copies for the "Risk Evaluation" team members.

3) IDENTIFY AND EVALUATE RISKS

This activity is at the heart of the risk analysis process. Its aim is to translate subjective individual opinions into a reasonably objective over-all evaluation. The diversity of your team and the combination of individual/group work are key to the process.

You'll essentially be using a Delphi technique -- combining individual evaluations with group interaction. The intent of the Delphi technique is to reach a group consensus, while minimizing the impact of personal dominance or differing verbal skills.

** At the first meeting:

-Explain the risk analysis process, the schedule you've set up, and how the results will be used. Explain that you've chosen the group members for their variety of viewpoints, and that you appreciate their involvement.

-Give them a copy of the asset inventory form (RA-1). Review it so that they will understand the importance and extent of your computer installation.

-Hand out copies of RA-3 and RA-4.

Tell them that you do not want their verbal evaluations at this time. Instead, they are to take the forms and work on them individually after the meeting.

Field any questions. Agree on times to get the completed RA-4 evaluation forms back from them, and to meet again.

** Between the first and second meetings:

- Collect the completed RA-4 forms from everyone. (Have them keep a copy for themselves.)

- "Quantify" their answers. This involves assigning numbers to their answers, and multiplying the "probablility" entry by the "severity" entry. For example, a "slight" probability and a "medium" severity would translate into 2 x 3 = 6.

Do these calculations for every entry on each form.

-Consolidate the evaluation numbers from all team members. Compute the average and the range for each risk item. Afterwards, put the risks in a sequential ranking based on their combined averages (i.e., highest average numbers at the top and lowest at the bottom). After all the evaluations have been consolidated, make enough copies for the team members.

** At the second meeting:

Share the results. Go over the key items, and allow people to explain the reasons for their ratings. This is an important sharing of data, since the diversity of the group can bring up valuable information about vulnerabilities.

In the end, there should be a general agreement about the sequential ranking of risks at your data center. Have them individually pick the 4 most significant risks that should be addressed. These will become your "A" priorities.

Identify a cut-off line for "C" priorities, based on improbable /unimportant events. You're left with a middle range of risks, which count as "B" priority problems. You want to focus most of your attention on resolving the A's, and plan on merely accepting the risks for the C's.

(It is recommended that you keep the risk evaluation worksheets that the team members prepared. With this source data and the subsequent rankings/alternatives/ implementation schedule, you will be able to demonstrate to auditors that you went through a thoughtful and thorough process in analyzing risks.)

4) BRAINSTORM OPTIONS FOR REDUCING RISKS

In the same meeting, you will lead a brainstorming session. The goal is to make a full list of things that could be done to reduce the risks. This information will augment the items already suggested on the RA-4 forms.

It's useful to write these ideas on an easel/blackboard during the meeting, so the group can see what alternatives they've developed. Seeing the list often stimulates other ideas.

Emphasize that you are not evaluating these suggestions at this time--evaluation comes as a separate, later step. You are now just trying to develop the largest possible list of things that could be done. (Even if the ideas are somewhat zany.)

5) RESEARCH COSTS/BENEFITS AND EVALUATE ALTERNATIVES

After the preceeding meetings have finished, one individual will be responsible for doing research. Essentially, this involves discovering approximate costs for implementing the various suggestions. Both out-of-pocket and manpower costs should be considered.

Center the majority of your time on the items with the highest total risk factor/priority. In the end, list your recommendations in priority order and review them with the group. This is the time for critical evaluation, and deciding on specific recommendations for implementation.

To handle the risks identified earlier, your basic choices are:

* to reduce the probablility of "bad things happening" by implementing one or more of the alternatives (Prevention)

* to develop a workable "disaster plan" that will insure a prompt return to normal processing (Recovery)

* to consciously choose to accept certain risks (Acceptance) (Usually this happens when there is a relatively low level of risk, or when the costs of implementing "countermeasures" are considered too high.)

(Obviously, accepting some risks is an integral part of life. The key point is that there is an evaluation and conscious choice about what will be done with the risks that are recognized.)

6) MAKE RECOMMENDATIONS AND SCHEDULE IMPLEMENTATION

This last step is the "wrap up". The information that has been gathered is presented to the appropriate level of management for decisions and action.

In your recommendations, remember that management will be looking for options that appear to be the most effective and offer the best return for the resources invested.

By the end of the presentation, decisions should be made on what will be done -- to minimize the risks and to protect the division's data-processing assets. An implementation schedule will be developed, responsibilities will be assigned, and decisions will be documented.

Now that you've figured out what is an appropriate level of protection, the next step is to make it happen. Here's where your management skills come in, and the risk analysis method leaves off.

Site Evaluated: _____ RA-1
Date: _____

IDENTIFYING DATA PROCESSING ASSETS

ASSETS	REPLACEMENT VALUE	REPLACEMENT TIME UNTIL WORKING AGAIN
THE DATA CENTER:		
*Computer hardware, CPU and all peripherals (Indicate type and number of systems):		
.....	.....	.....
.....	.....	.....
.....	.....	.....
Datacommunications (phone lines, modems)	.....	.....
Terminals (CRT's) # in the data center: ____ Total # in the division, hooked up to data-center computers: _____	.....	.....
Off-line equipment (e.g.,data entry, microfiche, etc.)	.....	.....
*Facilities		
Air conditioning eqt	.....	.....
Electrical wiring, motor generator sets, etc.	.....	.....
Raised floor and other room specifications	.....	.....
*Related assets		
Office equipment, including desks, typewriters	.....	.....
Forms/scratch tapes /supplies	.....	.....
** TOTAL REPLACEMENT VALUE FOR THE DATA CENTER:	.....	
** THE CRITICAL PATH: TOTAL TIME UNTIL EVERYTHING IS RESTORED AND WORKING AGAIN, IF THE ENTIRE FACILITY NEEDED TO BE REPLACED		.....

PERSONNEL:

Number of people in operations staff: .....

Number of systems programmers: .....

Number of applications' programmers/analysts: .....

Number of people employed in the entire facility: .....

BUDGETS:

What is the amount budgeted for data processing services
for the current fiscal year: .....

What is the estimated gross annual revenue for the entity
for the current fiscal year: .....

APPLICATIONS:

Number of major production applications .....

Functional areas affected by the applications .....

.....

Which applications are on-line? .....

.....

INSURANCE:

Insuring company.....

$ Amount of coverage for this facility is: .....

Items covered: .....

Items excluded: .....

SOFTWARE/DATA PROTECTION:

Number of dial-up lines (for "outside access"): .....

Number of people/groups with "priviledged mode" authority: .....

Indicate by type of hardware if there is a
system to control access to software/data:
(such as ACF2/RACF/Topsecret on mainframe computers)

....
....
....

Are data/software assets copied to tape and
preserved away from the data center? .....

Location .....

Average number of tapes.....

The frequency of storage is: .....

The length of retention is: .....

EXISTING SECURITY

What security measures are already in place to provide protection
for data-processing assets?

.....

.....

.....

.....

RA-2

TYPES OF PEOPLE TO BE INVOLVED IN THE
RISK EVALUATION PROCESS

DATA PROCESSING

Operations manager
Shift supervisor (the off-shifts tend to carry more risk
because there are less 'resource people'
around to help when problems occur)
Systems programmer
Hardware technician
Contracts specialist (for costs, and lead-time)
Input/output/forms supervisor
DP security analyst
Group leaders from 1-2 major applications

OTHER DIVISION PERSONNEL

HP insurance specialist ("loss prevention")
Security service (guards) manager
Building maintenance people (day-to-day repairs)
Building engineers (designing facilities changes)
Others (especially individuals who deal with 'outside' support
personnel, such as telecommunications,
electricity, and fire/police services.)

* NOTES *

--This list is meant as a suggestion for types of people to involve. In some DP installations, one individual may combine several specialities and functions.

--The group size should ideally be 6-12, in order to get an adequate statistical sample.

--The diversity of the group is its greatest strength. The perspectives of non-data processing personnel can be especially valuable (even though they may not "speak" in the DP "language").

RA-3

** DEFINING TERMS FOR RISK EVALUATION **

PROBABILITY:

-(Virtually) None — Extremely unlikely to happen
Example: Tornado in San Francisco

-Remote — Chance of occurence is quite improbable
Example: Stray aircraft crashing into the building

-Slight — Although infrequent or unlikely, it could happen
Example: Major earthquake in San Francisco

-Possible(medium) — A reasonable chance of occurance
Example: Fire, where smoking is allowed in the computer room

-Probable(high) — Very likely to happen
Examples: Programming error
Temperature/humidity fluctuations
Accidental mishandling of eqt

SEVERITY OF LOSS:

-(Virtually) None — So small as to be inconsequential
Example: Petty theft (of a terminal)

-Minor/low — Small monetary loss with little impact
Example: Power interrupt
Extremes in temperature/humidity

-Medium — Moderate inconvenience, or an expense for replacement of equipment
Example: Printer fire
Roof leaking water

-Serious/high — Severe losses and major inconvenience
Example: Major CPU fire

-Catastrophic — Huge negative impact. Restoration time could take weeks or months. Major $ losses.
Example: Bombing/sabotage
Aircraft crash

Name: _____ RA-4
Date: _____

** RISK EVALUATION FORM **

Listed below are some potential events that could disrupt data processing operations. Please respond to each item according to your own best judgement of how it might apply at your location.

The main options for the specific items are summarized here. For a fuller explanation of the ranking options, see attachment RA-3.

"Probability" choices		"Severity" choices	
(Virtually) None	=VN	(Virtually) None	=VN
Remote	=R	Minor/low	=LO
Slight	=S	Medium	=M
Possible (medium)	=PS	Serious/high	=HI
Probable (high)	=PR	Catastrophic	=C

Possible Disaster	Probability of occurence	Severity of loss	Comments/Notes
Extended computer downtime(no destruction)			
**Mainframes	_____	_____	
**Mini-computers	_____	_____	
**Data transmission	_____	_____	
Fire in computer room			
-staffed--weekdays	_____	_____	
-staffed--off-shifts, weekends	_____	_____	
-when not staffed	_____	_____	
Fire/explosion in the utilities room	_____	_____	
Fire in computer room and off-site vault at same time	_____	_____	
Explosion/fire in other areas of the division that could threaten data processing	_____	_____	
Water Damage			
-Flood	_____	_____	
-Roof leaking	_____	_____	
-Sprinklers	_____	_____	

	PROBABILITY	SEVERITY
Earthquake (severe)	_____	_____
Power failure or interrupt	_____	_____
Data-communications outage	_____	_____
Temperature/ humidity extremes	_____	_____
Lightning and storm damage	_____	_____
Tornado/Wind/ Hurricane	_____	_____
Damage by vehicle/ aircraft	_____	_____
Roof collapse	_____	_____
Damage/theft/loss of special forms	_____	_____
Vandalism/theft		
-internal source	_____	_____
-external source	_____	_____
Violent Sabotage (bombing, riot)	_____	_____
Non-violent sabotage (e.g., from a disgruntled operator or programmer)	_____	_____
Illegal access, and data compromised		
-internal source	_____	_____
-external source	_____	_____
Off-site tape storage vault: (assuming one exists)		
Fire	_____	_____
Theft/Vandalism	_____	_____
Sabotage	_____	_____
Other(pls specify)	_____	_____

Other possible risks--list your own:
(sources of potential disruption to the data center)

THREAT	PROBABILITY	SEVERITY
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

****

On a scale from 1-10, what ranking would you give for the over-all level of risk at this site? (1= Very Low, 10= Very High)

_____

On a scale from 1-10, how important would you say that this computer facility is for the survival and profitability of the company as a whole? (1= Very Low, 10=Vitally important)

_____

****

GENERAL QUESTIONS:

1) What would you say are the most likely possible natural disasters in this geographical area? (e.g., tornados, earthquakes, electrical storms, etc.) _____
_____
_____

2) What things tend to keep the risk-level low at this site?
_____
_____
_____

3) What things tend to increase the level of risk at this site?
_____
_____
_____

4) What changes in the future might affect the level of risk?
_____
_____
_____

5) The Community Environment (safe or hostile)--

What is the role and image
of the company in the community? _____

_____

How would you describe the social safety of the environment?
(Consider things like--Are cars/houses normally locked?
Are there universities or "radical environments" nearby?)

_____

_____

What is your over-all evaluation of the probability of
a threat from a non-employee? _____

_____

****

6) What preventative things do you think we might do to reduce the level of risk (and therefore to reduce the chances of a disruption in computer processing)?_____

_____

_____

_____

_____

_____

_____

_____

CENTRAL SITE SUPPORT: A CASE STUDY

by
Richard Diehl
of
Monsanto Industrial Chemicals Company

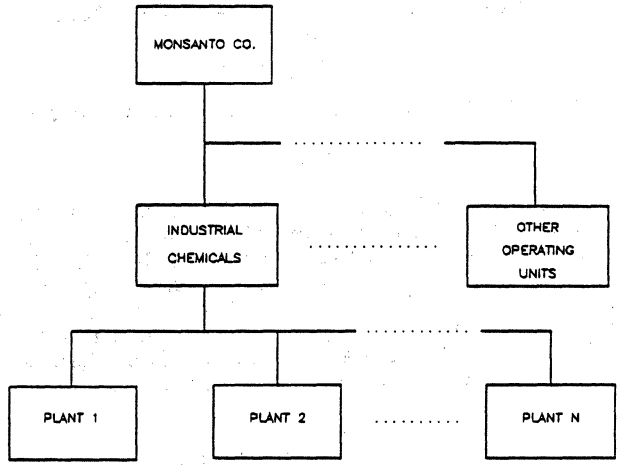
Introduction

Monsanto has many HP-3000 installations throughout the company. Some are standalone sites with their own CSS support. Some are supported centrally by the Industrial Chemicals central site. The election to go to central site support was initially a financial one. However, procedures, standards and people had to be put in place to make it work. This paper deals with what we've put in place and the unforeseen advantages we've gained from some of them.

Introduction to Monsanto

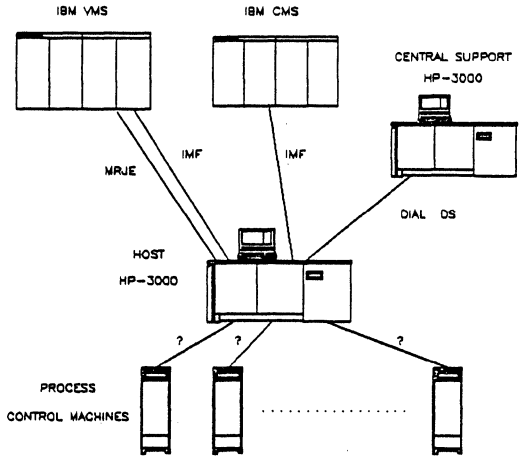
Monsanto's central site support has not evolved in a vacuum. It has come into being to answer specific needs of Monsanto Industrial Chemicals Company. These needs are largely generated by Monsanto's business and its organization. Therefore, any discussion of the Monsanto central site support must begin with an overview of Monsanto.

Monsanto is a large, multinational chemical manufacturing company. Its products range from soap to silicon chips. To adequately manage our large and diverse product mix, Monsanto is divided into several semi-autonomous operational units (see FIG 1). Each operational unit has responsibility for its own MIS. Therefore, within Monsanto we have networks of DEC, IBM, and HP equipment. The op unit that I am associated with, Industrial Chemicals, has selected HP as its preferred vendor. This means that the data processing computers within the Industrial Chemicals plants are HP-3000.



MONSANTO ORGANIZATION FIG 1

In reality Industrial Chemicals is just a collection of plants. Each plant has a central data processing computer called the plant host. The host can be either a 3000/44 or a 3000/40. The size of the plant determines the type of machine and the staffing of the DP department. In our larger plants the host is a 44 and is managed by a data processing manager. Working for this manager is one operator, and in some plants one programmer. In the smaller plants the 40 is our host and the operator is an accountant with special training. We designate these smaller plants as unmanned sites. By unmanned we do not mean that there are no operational personnel on site, the accountants perform this function. Instead we mean that the normal decision and design duties of the system manager is performed by a person in the central support group. In addition to our host computer, the plant can contain special purpose computers used to control the running of the plant processes. These process computers are tied into our host so that data on inventory (quantity and quality of product) can be entered into Monsanto's data environment. In addition, the host is connected via MRJE and IMF into the corporate IBM systems (see FIG 2). On these large IBM machines run our traditional, corporate-wide data processing applications such as general ledger, accounts payable and receivable. Therefore, our host 3000 fits into an overall corporate-wide computer network and performs a prescribed function in Monsanto's computing environment.



COMPUTER NETWORK FIG 2

The charter or function of the host (HP-3000) is to provide on-line interactive processing of data that is relevant to that particular plant site. Typical applications that fall into this category are laboratory information systems, maintenance control systems, storeroom management systems, and employee time card systems. If the data generated by a system is required by production applications outside the plant, such as inventory levels for marketing, then the system resides on the corporate IBM machines (see FIG 3). Also we have systems that run on both machines, front-end editing and plant specific data bases are handled by the HP, and the updating and inquiry of corporate-wide data is handled by the IBM. Each plant uses only the applications systems it can justify. This puts the burden on our central support to be able to handle a wide variety of systems.

APPLICATION	PRIMARILY HP		MIXED		PRIMARILY IBM	
	BATCH	INTERACTIVE	IMF	MRJE	IMF	MRJE
PAYROLL						*
INVENTORY			*			
BUDGET						*
CLOSING						*
STORES MGMT.		*				
LAB INFO		*				
TIME CARDS				*		
MAINTENANCE MGMT.		*				
IBM DEVELOPMENT					*	
PROCESS COMPUTERS	*					

HP JOB DRIFT →

← IBM JOB DRIFT

HOST APPLICATIONS FIG 3

Our central support 3000 is a 44 used by the central support group for development, systems support, and production for the central staff organizations such as division accounting, personnel, and marketing. It is also tied into the corporate IBM machines via MRJE and IMF. In addition, our plants are tied to the central 3000 by a 4800 bps dial-up, DS network. The DS network is not used for production data processing but is instead reserved for systems and applications maintenance.

Central site support description

Our charter for the host and the functions of our central machine came well in advance of our selection of the vendor. We were aware of the advantages of central support because of our experiences with IBM and wished to avail ourselves of these advantages with our plant hosts. Briefly, these advantages are cost savings, central vendor interface, and standardization. The cost savings are obvious and the reason we can sell the central support to the plants. I will deal with it shortly. The next two advantages first appear as disadvantages. These must be done to make the central site support work. After running awhile, the advantages of central vendor interfaces and standardization will become evident. The bulk of the remainder of this paper will deal with these latter advantages.

However, let me first describe what is meant by Hewlett Packard central site support. It is a method of supporting multiple HP sites as if they are one site (as far as HP is concerned). The HP SE office deals with

only one site, our so called central site. All system updates are delivered only to this site. It is the responsibility of Monsanto to distribute the update to the 2'nd sites. Since we have CSS support we also receive PICS, on-site SE assistence, and manual updates. Manual updates are sent only to the central site and only in a sufficient quantity to update the central site. It is the responsibility of Monsanto to order additional updates and get them to the 2'nd sites. PICS calls can only be made by the system manager of the central site (unless we have purchased additional PICS caller support). Therefore, all problems must be routed through the central site. Likewise, the SE assistance is given only to the central site. If an SE visit is required by a 2'nd site then it must be purchased on a time and materials basis or a Monsanto employee with enough training and experience needs to be sent.

There are variations on this plan. For instance we have found it useful to provide additional PICS callers at those plant sites involved in heavy applications development work. This service is fairly inexpensive ($100 a month) and provides our programmers with a quick answer to his or her design or implementation problem. Trying to keep Monsanto employees adequately trained on all the HP systems is just too great a strain on our organization. Therefore, we prefer to hire out the detailed implementation consulting and concentrate on the systems support issues.

We have looked into central distribution and manual update service for our sites but do not consider them cost effective. Central distribution is where our HP sales office would mail out systems tapes to all our 2'nd sites. Manual update service is where HP will supply additional manual updates to our 2'nd sites. With proper stream job setup, the cutting of systems tapes can be done by an operator, and the manual updates can be handled by a secretary. Therefore, it costs us very little to handle these tasks.

However there is a cost involved in central site support. Our 2'nd sites are our customers and we must provide them with a level of support that keeps them satisfied. If we don't, they have the option of purchasing HP CSS support (the plants are autonomous entities). In actuality, we attempt to do a better job than HP. We support our sites at the Monsanto applications systems level, not just the HP systems level. Therefore, most of our problems first come in to analysts as applications problems. They are our front line consultants and our direction is to make them as productive as possible. The central site systems manager backs them up as well as handling questions

that don't fall into a given applications system. Hence, both the analysts and the system managers must have knowledge of the HP. This additional learning and workload on the analyst must be recognized, and specific steps taken to lighten the burden. All this will impact what at first appears to be the cost savings involved in central site support.

Central site cost analysis

The main advantage in central site support comes from software maintenance charges. Each site must purchase the software initially. If your central site is the first site in the organization to buy a particular software package, then they will have to pay full price for the software. All other sites buying the software will pay the reduced right-to-copy price. This may or may not affect your central site cost analysis because you may not be buying the first copy for your organization. In Monsanto this does not figure in because it is rare that our central site must purchase the first copy. The following table gives the difference in monthly service costs for the central site and the remote sites for a typically configured system.

Software	CSS cost	2'nd site cost
FOS/40	$350	$70
DS	$125	$35
MRJE	$ 75	$15
IMF	$125	$35
COBOL II	$100	$25
DSG	$ 65	$30
Total	$840	$210

Total savings per site per month $630

The cost involved in maintaining the support for the 2'nd sites is dependent on what level of support is given each site. As I noted before, we at Monsanto provide system update tapes, and manual notification service. In addition, we provide consulation on system configuration, systems error handling, and development consultation. The first two are easy to quantify. It requires half an hour of operator time per update tape, per site, for the system updates. The manual notification service takes about 15 minutes of secretarial time per update (by using mass mailing we can handle many sites for the same cost as one).

The consultation services and system error handling are the most time consuming and the hardest to quantify.

Consultation problems fall onto a continuum of easy to complex. The easy problems are those that can be answered quickly from the consultants experience or by referencing a manual. These take about 5 minutes of the consultants time. The complex problems require extensive research and testing and often involve consultation with HP. These can take weeks of a consultants time. Therefore, we have a wide range of consultation costs - 5 minutes to man-weeks. It should be noted that there is a second order factor affecting problem consultation. This is the fact that as the people at a given site gain expertise, the problems reported for consultation tend to be of increasing complexity. In actuality we reach a saturation point where the complexity of the problem is such that we at the central site can only verify the problem and pass it along to HP for solution. In effect, the consultant becomes a liaison person with HP. So in estimating the average time taken by problem consultation, we need to weight the average to the high end of the curve, but not too far over.

Finally, to arrive at the cost of support, we need to estimate the number of consultation problems we have per site. When the site starts up we can expect an average of one problem per day for the first month and a half. This rate tapers off to about two problems a month for the mature sites.

Obviously there is a large variable cost associated with the support, namely that of consultation. For central site support to meet its cost savings promise, all the sites must be managed in such a way as to minimize this cost. Standardization can go a long way to holding down the consultation cost as well as yielding side benifits that may actuall outweigh the savings in consultation time.

Standardization

When a problem call comes into the central site it is usually of the form -- "Job B1753'S printout didn't come out. Do I have to rerun the job?". -- This presents the consultant with several problems, some of which are:

1. Is this an HP or an IBM job?

2. What production system is it associated with?

3. Where is the printout supposed to come out?
    a. Is it spooled on the HP?
    b. Is it also spooled on the IBM?

4. What files does B1753 create? Where are they located... in an IBM PDS? If so, what directory and dataset? Are they on the HP? If so, what account and group is it in? Are there files on both HP and IBM, and if so, where?

5. How can I look at the files on the HP with the problem (it may be located 2000 miles away)?

6. What questions do I need to ask to get the right information in the shortest time?

Our consultants would have to spend a large amount of time determining answers for these kinds of basic questions for each problem call without standardization in hardware, systems, and applications software. In addition, the solutions found for one site might not map into other sites. All our sites run about the same functions so problems encountered by one are usually encountered by all. Without standardization we would have to solve the same problem several times. However, if we standardize, we can solve the problem once and communicate the solution to all the sites. In some cases this allows us to head off problems before they happen.

In addition to the gains in consultation efficiency, standardization also allows us to develop plant specific applications at one site and then easily move them to another site. For instance, all our plants have a QC lab and all of them handle data in about the same way. We were able to develop a laboratory information system for one plant and when it was complete move it into the QC labs of 2 other sites within the same quarter. However, to gain the advantages of standardization we found that we had to standardize in great detail. Some general guidelines for applications development and systems configuration would not work. Therefore, we established standards in hardware, systems, and applications development. Hardware standardization was the first item we attacked.

Our hardware standardization was affected by our organization. As I mentioned earlier, the sites have few if any DP people. Therefore, when selecting a hardware standard, we attempted to lighten the management and tracking load on our site people. This boils down to the philosophy "Let HP do it". We are willing to accept an adequate piece of HP equipment where a super piece of 3'rd party vendor equipment exists at a lower price. We do this so that our site systems managers will not have to mediate hardware problems between vendors. We have standardized on the series 4x HP's with HP disks, tapes, printers and tubes.

By standardizing the hardware, we have been able to standardize our I/O configuration. This aids the consultant in problem handling as they are familiar with the configuration. It's the same as the one on their machine. In addition, this standardization has allowed us at the central site to establish starter kits. A starter kit is a predefined package of hardware and software needed to run an application. Therefore, when a plant manager asks us "How much will it cost for us to implement a host computer?" or "How much for a LIS system?". We can say XXXXX dollars. We don't have to convene a study team, make a preliminary survey, write a recommendation, go out for bids, and gain approval. We merely tell the plants the cost, give them the areas justification can be found in, and allow them to decide. Obviously approval must still be gained but a large up front study cost has been eliminated.

In addition to standardizing the HP hardware, we've nailed down our communications systems. Our DS dial network is Bell 208B modems. This was done again to ease the hassle our site people have with multiple vendors. If we have trouble with DS, the DSM program will checkout the link completely through the analog loopback on the 208B and then it's Bell's problem. Well it was. We're looking into this policy under the new Bell reorganization.

Our leased line network into our IBM machines have been standardized at a corporate level and are handled by telecom experts. We in Industrial Chemicals have pushed our standards back one level into the IBM systems. We've standardized our JES configuration and have standardized on a 1920 character buffer 3277 for IMF. In addition to aiding our consultants, they don't have to worry abour debugging IBM and HP systems when one of our joint applications runs into trouble. It has simplified startup immensely. When we start up a new 3000, our IBM systems programmer copies the macro from our central site JES gen, changes the remote number and we've got a new remote. Likewise, our telecom people know what modems and strapping is required. This has shortened our bring up time to 1 day for systems installation and telecom startup.

The standardization of the telecom and the hardware provides us the underpinning on which we can standardize the system configuration and software. In the systems configuration we have followed HP's lead. We've placed our reliance on class names not LDEV numbers. We have the standard classes of devices - DISC, TAPE, SPOOL, DDUMP, and LP, as well as our own classes - TERM, MODETERM, DSCLS, INP, MRJE, IMLPSUDO. All of our systems are constructed to utilize the device class

names whenever possible. Our consultants, when faced with a problem, do not then have to be aware of what LDEV number the particular unit is. They merely have to know if it is the proper class. Also our analysts can set up jobs that can easily be moved from one system to another.

In reality the standard system configuration is easy to setup. Again, judicious use of a job stream running SYSDUMP with embedded answers to the configuration questions allows our operator to quickly cut the initial systems load tape in a short period of time.

The first step taken to standardize the software was to standardize the system SL. Even though each site does not get every HP product, each site gets the same system SL. This allows us to easily install new products at a site. We merely copy in the PUB.SYS files. In addition, we allow no second party segments in the system SL (either another vendors or our own). We can then avoid all SL maintenance activities. In other words, our sites don't have to run SOFTWARE.CREATOR.

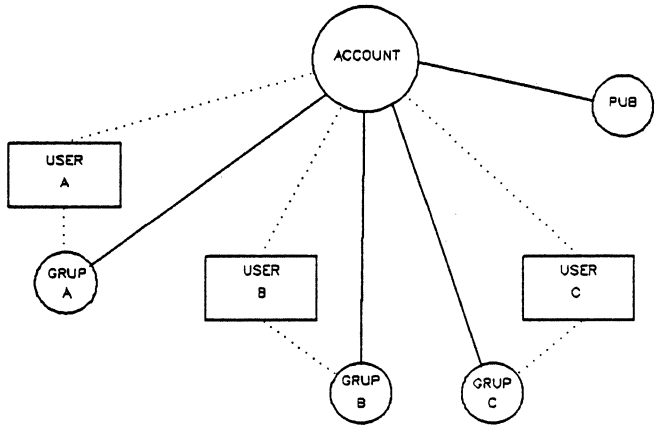
In addition to our SL, the PUB.SYS group and the HPUNSUP.SUPPORT group are kept free of non-HP files. Any systems-wide utilities developed by Monsanto or from the IUG are stored in the group UTIL.SYS. This division simplifies the consultants job when they are faced with a program problem. If the program resides in PUB.SYS, look in the SSB. If the program resides in UTIL.SYS, call Monsanto's central site system manager.

This practice of keeping only HP supplied software in PUB.SYS was arrived at by common sense. However, the standardization of the accounting structure evolved over many months and involved much debate. Basically we have classified accounts into three classes -- system or vendor accounts, group accounts, and development/production accounts.

A system or vendor account is an account whose structure is defined by HP or a vendor. We have no control over its structure. However, we do not allow users onto these accounts. The only I.D.'s on these accounts are those required for maintenance and these are password protected with the system manager holding the password. We allow access to these accounts by users of the system only to the level required to run the system.

The group accounts are used for personal computing. These accounts are structured with one user per group and one group per user (see FIG 4). All groups are password protected denying logon access to the other

users. Each user logon ID is password protected. These accounts are used by people that wish to use the HP as their personal tool. The typical user runs spread sheet analysis or develops small programs to help them at their jobs. With the growing awareness of computers and their use, the demand for this type of account is growing. While a user may use his or her own developed programs for day-to-day work activities, we do not allow formal production to be run on one of these accounts.

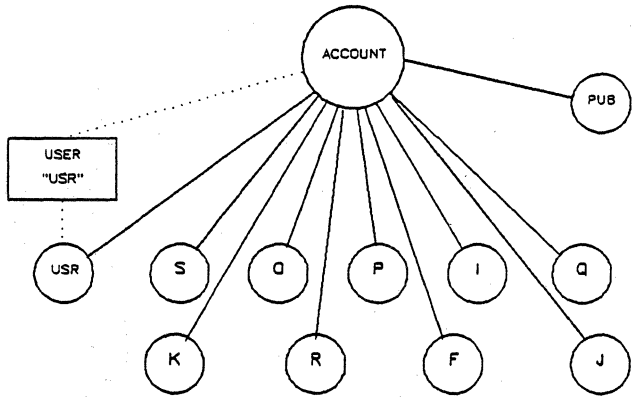


GROUP ACCOUNTING STRUCTURE FIG 4

Production systems are those where a program or system of programs are run by designated operational personnel on a regular basis to support a formalized information flow. To support this we have adopted the production/development account (for simplicity I will refer to these as production accounts). The production account has a functional group structure. There is one group for each type of file P for programs, O for USLs, S for source, I for data bases, K for ksam, Q for standard MPE data files, R for reports, F for forms files, and J for jobs (see FIG 5). The PUB group is reserved for account and user UDC files and for SL's associated with this system. In some cases several users will be simultaneously creating files with the same name. Usually this occurs using ENTRY. In this case, each user is given his or her own group -- essentially a private Q group.

The security for these groups is dictated by what each group does. The access is as followes:

Group	Access
PUB	X,R,L:AC;W,A,S:GU
P,J	X:AC;R,W,L,A,S:GU
O,S	R,W,L,A,X,S:GU,AL
I	R,W,L,A,X,S:ANY
Q,R,K	R,W,L,A,X,S:AC
F	R,L:AC;W,A,X,S:GU,AL



STANDARD ACCOUNTING STRUCTURE FIG 5

The overall account security is R,W,L,A,X:AC if no sharing of data in an image data base is required. If the data from a data base is needed outside the account, then the security is R,W,L,A:ANY;X:AC and access security to the data base is handled by IMAGE.

The user either logs on to the Q group, I group, or their own group depending on where most of their files are located. The programmers are given account librarian status and their home is usually S. The account manager controls the P, J, and PUB groups, and therefore, can control the movement of programs, jobs, and UDCs into production.

By standardizing the accounting structure, we gain efficiency in the problem consulting area because the questions normally asked about security and capabilities have been answered in advance by the standard. In addition, we gain in programmer/analysts productivity because they don't have to worry about designing the account structure or spend time consulting with the system manager as to the proper structure for each system. Finally, the portability of systems are enhanced. We don't have to debug a seperate accounting structure on each machine. We can STORE from one CPU and RESTORE onto another CPU.

The standardization of the production account is taken a step further. The file names within the production account are standardized. The standards are based on a three letter system ID. This ID is unique for each applications system and is used to preface many of the files. It gives us the ability to group all files for a given system when they are archived on the IBM librarian. The naming standards are as follows:

Type of file	Form name
Image data base	SSSNNN
MPE or KSAM file controlled by a program	PXXXAAFF
MPE or KSAM file controlled by a job	JXXXAAFF
Jobs	JOBNNN
Programs	SSSNNN
Form files	DDDDDDTN

Where:

SSS	is the system ID.
NNN	is a sequentially assigned number
XXX	is the sequentially assigned number for the corresponding job or program.
DDDDDD	is the ID for a program, job or data base.
T	is the type or form - I for image, K for KSAM Q for sequential, P for programs.
FF	is the format number used to reference the file format in the documentation.

The naming conventions allow not only the consultant to quickly determine the file dependencies within a system, but also allows management to easily move analysts from system to system. The analysts have only one set of names to learn, and they can be applied to any system, either the new one in design or the one that is having emergency maintenance performed on it. The analysts don't have to consult possibily out-of-date documentation to fine what file is what.

If we've developed standards to this level of detail, it would be probable to suppose that they exist for design, coding, and documentation, and they do. However, I don't have the space to do them justice. It is sufficient to say they exist in as great a detail as the accounting structure examples given above and aid us in design and maintenance.

Therefore, we can see that the consultant faced with the question -- "Job B1753's printout didn't come out. Do I have to rerun the job?" -- would know:

1. The job name didn't start with J so it is an IBM job.

2. It is associated with standard 1700 systems production (per our IBM standards, that I haven't outlined).

3. Printout should have come through the JES spooler to the HP spooler and been routed to a 2608 printer class LP.

4. All files are located on the IBM and are catalogued as per our IBM standard.

5. No need to look at HP (it's usually not at fault anyway).

6. Call Monsanto customer service for a restart.

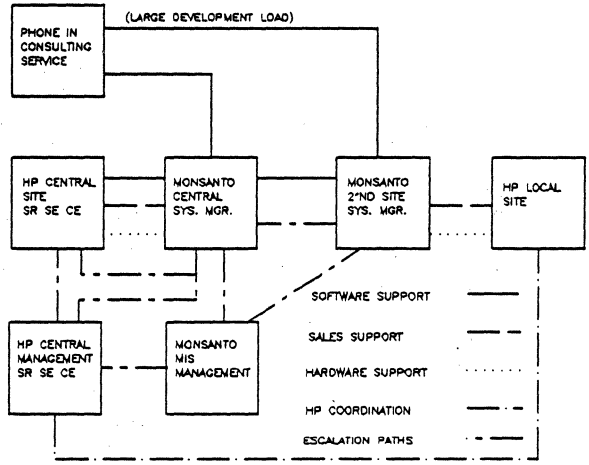
Central vendor interface

As was mentioned earlier, one requirement placed on us by HP for central site support was that HP deal only with our central site for questions involving software. Each of our sites maintains its own hardware service contract with the particular HP local office in their region.

The fact that there is no local SE for the CE to turn to when he suspects what was called in as a hardware problem is in reality a software problem was recognized early to be a potential problem. In addition, the lack of SE involvement at the local sites causes the normal HP problem escalation procedure to behave in an unusual way, if at all. Internal to Monsanto we perceived a problem in picking up the duties of the SE; namely, what would we do about backing up our central site systems manager?

The central site systems manager is the contact for HP for all problems and the contact for the sites for SE support. When we purchase CSS support from HP we are assigned a SE but in reality we are buying the services of an SE organization. If our SE is sick or unavailable, HP provides a backup. Monsanto was faced with a need to provide backup for the central site system manager. Having a backup to the central site systems manager could cause us communications problems with HP. Should the SE accept problem calls from the backup? Or more basic, who is the backup?

It was obvious as Monsanto and HP began discussing the details as to how the central site support would work, some formal definitions would have to be drawn up. This was done by our sales rep in the form of a detailed document called Monsanto's Central Site Support Plan. It detailed the communications paths for hardware and software problems within HP and the contact points between Monsanto and HP for hardware and software. In addition, it detailed what each party could expect from the other. The essence of this plan is shown in our support diagram (see FIG 6). In essence, our central site systems manager, or backup, deals with the St. Louis SE on problem reporting, definition and resolution. The central site systems manager or his management can request escalation for either hardware of software problems from the St. Louis SE, sales rep, or SE or CE management. HP would put in place the necessary communications to the other HP regional offices and Monsanto would handle its own internal notification system.



MONSANTO CENTRAL SUPPORT PLAN FIG 6

To define Monsanto's internal system, including backup we took the HP developed support plan and expanded the areas detailing Monsanto's responsibilities. We provided procedures, problem classification, routes for escalation, and backup lists. This complete plan, Monsanto's and HP's, was then published to the sites and it has become our working document for providing software service.

While this definition of central support was required to enable us to utilize the central site support from HP, it has since provided Monsanto with advantages over

and above the cost savings involved in central site support.

The most obvious savings is a central clearing house for all problems within Monsanto. If one of our sites has a problem, say with IMF, we are able to inform our other IMF sites to watch out for this problem, and in some instances give them the workaround before the problem causes them to loose any productivity. The central clearing house also aids in problem definition. If there are two sites with the same MRJE problem that eliminates the modems, line, INP, CPU and memory. This just from the knowledge that the problem occurs at two of our sites. We can then concentrate on determining if we have problems with the 3705's or JES, and HP can concentrate on the software. This savings may at first seem trivial, but about 30% of our problem solution effort involves telecommunication systems, and it takes an elapsed time of one day to test the modems and line. Therefore, having another almost identical site to compare with can save us a day's time.

The central clearing house also save us time in systems development. Problems are reported to the central site and they are tracked down as being either HP problems or problems with applications. In either case the problems, and solutions or workarounds are known. This pool of information has proven useful when evaluating designs for applications systems. We can identify practices that cause operational problems, known bugs that will impact the design, and techniques that can increase the likelihood of problems (such as interactive use to the telecom lines).

An obvious extension of our central site support for HP is to include other vendors. We have found that some vendors are willing to consider this type of support for multiple sites (we would like to see them all do it). This gains us all the above mentioned advantages plus one other.

I've alluded before to the problem of having small or no DP staffs at our sites. Like most companies, we rely heavily on our computer and the use of the computer is increasing. This forces an increasing workload onto our DP staffs. This workload has shifted in recent years from operational to support. As our batch systems are replaced by on-line systems and as personal computing increases, our DP staffs are having to move from operations into training and consulting. Both of these are labor intensive, time consuming functions. Anytime a new person is hired or a clerk is sick, the DP supervisor finds him or herself out in that operational department for long periods of time. This prevents him

or her from being available for other problems. In addition, systems are becoming more complex and the amount our DP supervisors need to know is increasing to such a level that they can't remember where to find the data, much less what the data is. Obviously, steps need to be taken to alleviate this problem, and our central support can help.

By taking over central support for HP and vendors and because we are employees of Monsanto, we can lift some of the burden off the DP supervisors. They no longer have to call this number for HP problems, that number for TESS, this other number for MICOM. All they need to do is call central site support. We at the central site will sort out what vendor to contact. In addition, if the problem is with an applications system we can get the user at the plant in touch directly with the analysts supporting that application relieving somewhat the handholding burden of the site DP personal. This can save us staffing at our remote sites, thus lowering our DP overhead.

In reality, it is a reduction in needed staff, not just a transfer of staff duties from the plant to the central site. If a support task takes .3 people at a site, it may take .2 people per site if centralized. This occurs because of the learning curve of the people involved. Our .3 people spread over 3 sites would account for an increased headcount of 1, but in reality each person at the sites would spend 70% of their time doing something else. They would have to relearn their duties everytime a problem occurred, hence the estimate of .3 people for support. By centralizing, we can afford to devote 100% of a person's time to this system. This person would not have to reclimb the learning curve for each occurrence of the problem, and by increasing his or her efficiency in this way one person could support 5 sites. This is an old argument ... Centralized vs. Decentralized ... and the trick is to manage the trade-off well enough to pull it off.

There are some things that can be centralized and some things that cannot. By standardization and proper tracking techniques we can easily centralize the vendor interface and gain efficiency, freeing the site DP supervisor to handle the unique plant specific problems that should not be centralized.

One surprising area we've been able to centralize in our vendor support is that of sales. We've been able to handle contract negotiation, systems configurations, and order expediting for the central site. By doing this we not only off-load some of the work of our DP supervisor but also are able to insure standardization

in the hardware and software. In addition, we can exert a greater leverage over the vendor. Our central site can speak for several systems not just one machine. The vendors take more notice of selling 10 systems than 1. At least we believe they do, and our experience with HP, IBM, and DEC to-date indicates the mainframe vendors do.

By centralizing all our vendor interfaces at one site we've placed a large burden on that site to communicate with the other sites. We have found that we must behave very much like a vendor's SE staff in keeping our customers happy. The pitfall of assuming we're all Monsanto, and therefore the central site interests are the same as the plant sites is easy to fall into. When this happens the plants feel isolated and dictated to, and then the them vs. us mode of operation starts. This will cause the plants to waste energy on non-productive avenues and undercut the basis on which central support gains advantage. We are constantly finding areas where this is occurring and, I hope, correcting them.

We have instituted the site support survey. This is a questionnaire sent out twice a year on which the sites rate our central staff on their performance and the plants can make suggestions for improvements to be made. This not only serves to point out our weaknesses but also makes the plants feel that they have a say in controlling their support. From this questionnaire we have discovered two areas that we have taken special pains to remedy. One is problem tracking; the other is news of what's going on.

The news issue was solved by publishing a monthly newsletter and throwing it open for additions by anyone. This letter allows us to describe what will be coming out in the next release, what new products are under evaluation, and what major bugs exist and where they are in the solution process. It provides management with a vehicle to list responsibility and personnel changes. It provides everyone a forum to exchange gossip about the HP systems, and the work being done. While the active participation by the plants has been small, the feedback we've been given indicates that just the publication of such a newsletter adds to the feeling of community among our HP sites.

The area of problem tracking was a little harder to solve. It was necessary to log all problems and log the actions taken to solve them, along with the results of those actions. We needed to produce reports of these problems for our management and send a copy of the appropriate problems to each site. In addition, we need a way to formally inform our vendors of serious problems

and track their responses. Finally, we felt to gain maximum benifit from central support a historical record of such problems needed to be kept. We could have instituted a manual system to track our problems but instead elected to automate the system and place it on the HP.

The system is interactive. Anyone at a site (via DS), or the central site can enter a problem or report on any action taken. The problems are assigned a status as they flow through the tracking process. The status is controlled by the central site system manager. Our automated job scheduler fires off status reports to management on all problems on a weekly basis. Also, the scheduler fires off a report of all problems for each site on the 10th working day of each month. These reports are routed to the plants via MRJE. In addition, anyone can interactively get a report on any given problem via a CRT tube. The central site system manager can trigger the generation of an SRO for any problem. Once a month the closed problems are archived to tape, providing us with an information library on problem solutions, response times and resources used. The automated nature of the reports helps to make sure no problems fall through the crack. The site DP supervisors get their monthly reports regardless of how much action was taken on a problem. If they feel the action was inappropriate, they will let us know. Likewise, we at the central site know our actions are being watched so we strive to do better.

I have not mentioned training with respect to our support. This is because our central site people have no special training. They have had the standard HP courses in System Management, IMAGE, and Programmers Intro. Any expertise they have has been gained from experience. We've found to-date, it is unsatisfactory to train people in great detail about any vendors products. They tend to forget the detail in a short time. Instead, we find they learn more by attempting to solve the problems. True, this does lead to some inefficiencies, but it is the best solution we've come up with to-date.

While our central vendor interface was forced onto us by central site support, I feel it has presented us with many opportunities to improve the data processing environment at Monsanto. Instead of costing us something, it has actually added to the value of our cental site support.

Conclusions

In the cost analysis section I noted our savings for an average site would be $630 per month. Offset against

this would be our cost of providing the SE-like service to our sites. We've gone beyond the SE service at Monsanto and for good reasons. We've realized benfits from this stretching of our service that improves our whole DP operation. Therefore, our real savings per site per month is at least $630, and probably more, if we had some way of accounting for the effort saved by the use of standardization and the improved efficiencies realized by centralized vendor interfaces. We are happy that we've elected to go to central site support and will continue with it into the foreseeable future.

I would like to thank the personnel of the St. Louis HP office for their help in preparing this paper and for the use of the software and laser printer used to compose this paper.

Using OPT and APS for Performance Optimization

Ingrid DiTommaso
Hewlett Packard (Canada) Ltd.
Calgary, Alberta, Canada

I. Introduction

There are many factors that affect system performance. At the highest level, performance is the result of processing requirements imposed by the operating system and application software on a given hardware configuration at a given time. A common measure of this outcome is response time and throughput. Response time is the amount of time required to service a user request, and as such is greatly influenced by the amount and availability of resources that are needed to complete this task. If response time is not adequate, then the limiting factors must be identified and solutions must be explored and implemented.

Performance measurement tools, e.g. software monitors, may be required to achieve these objectives. The monitoring tools measure how each of the components of a computer system are utilized. However, nothing can be said about how efficiently these components are operating. One approach towards improvements in efficiency is "Selective Recoding". The focus is on identifying which portions of program code are most often utilized during execution, and on investigating them for gross inefficiencies and possible optimization. It follows that, from an overall point of view, frequently used applications or resource-intense programs are the best candidates for this type of tuning.

In this paper the use of Hewlett Packard's software monitors, OPT (On-line Performance Tool) and APS (Application Program Sampler) will be explored. After a brief review of the general performance variables, the applications of OPT and APS are discussed from a general point of view. Selected examples are used to illustrate how to use these tools in a given situation.

II. Performance Factors

Computational resources (CPU, memory and disc I/O) have often been named key performance variables. Each application will require a certain amount of each resource, and as long as supply exceeds demand, response time depends on the complexity of each transaction, given that no other blockage exists. It is important to realize that most computer systems have the ability to allow concurrent usage of processing resources (CPU, I/O devices etc.). It takes several users to benefit from this "multiprogramming effect", and there is some trade-off between the benefits of the this parallelism and the increase in operating system overhead.

Other important considerations in performance are delays incurred during the duration of a transaction because a critical resource cannot be obtained. Common reasons for this type of blockage are waits for a lockable resource (i.e. files), or for a system table entry. The latter can be circumvented by adequate configuration parameters. Waits are also introduced as computational resources become scarce (queueing delays). Last but not least the application itself must be mentioned as a major contributor to performance. Poor design, inefficient code and highly resource-intense applications are often responsible for poor response times.

III. OPT and APS: Which to use when, and why.

Both OPT and APS collect utilization data. While OPT considers the system as a whole, APS focuses on a particular application. In a most general sense, OPT could be considered as a system tuning tool, while APS is strictly an application tuning tool.

OPT provides statistics for overall utilization of computational resources as well as individual process and program behaviour profiles. OPT is useful in answering the following questions: What is the current state of "health" of a computer system? Is response time limited by a shortage of system processing resources? If so, could the system workload be changed such that I/O, CPU and memory usage could be balanced? Does a single application dominate the mix? Can file placement be improved? Can the workload be increased without impacting response time? In summary, OPT can be used to identify performance bottlenecks, characterize and tune system workloads, tune the performance of individual programs, analyze system table configuration and collect data for capacity planning.

APS provides the basis for program efficiency optimization, i.e. selective recoding. It aids in identifying where within a program optimization efforts would result in the biggest gain.

IV. Using OPT

Before discussing how OPT can be used, let's take a look at the data that this tool makes available. OPT reports information based on six main categories (contexts), with each having one or many detailed displays:

1. The Global Context displays provide an overall picture of CPU usage, memory utilization and disc I/O traffic as well as a summary of process (job and session) activity.
2. The Memory Context displays show how memory is being used. Included are an overall memory usage report, frequency distribution histograms for segment types (code, stack, data and extra data segments) and images of memory bank contents.
3. High level data relating to CPU states and memory management activity is given in CPU/Memory Management Context displays. The rate of process launches and preemptions, the mean and maximum time in each of the CPU states, and the current and maximum average CPU time of interactive transactions is displayed. Memory allocation requests together with resulting actions, memory management related disc I/O and Clock Cycle Rates detail Memory management activities.
4. The I/O Context reports break I/O activity out by device (disc, tape, printer). Queue length distribution statistics are available for disc devices.
5. Process and some program specific information is provided by a variety of displays in the Process Context. The CPU usage, memory requirements, process priorities and process states are summarized for the entire system. For each process a detailed display includes information on stack utilization statistics, process wait states, file usage, detailed working set data and stack marker traces. Queues on SIR's (System Internal Resource) can be requested from various screens.
6. The System Table Utilization displays report system table usage data.

Depending on the applicaton of OPT/3000, one or all of the available displays may be consulted.

OPT can be used on-line or in batch mode, with optional logging. Batch and logged output are limited to reports

on computational resource statistics. Although this type of data is very useful for trend analysis, it must be supplemented for optimization or troubleshooting purposes. Why? Because performance always has to be viewed in context with system workload. If it is not known which applications are running, if batch is active, and what response times are, then conclusions cannot be drawn. Let's consider the following scenario which resulted in identical CPU utilization data (i.e. no CPU idle time) for two different system loads. The first mix consists of on-line users and one batch job doing sequential file access, the second consists of on-line users only. Summary reports in both cases show that the CPU is fully utilized (no idle time). Can we conclude that we have run out of CPU power based on these reports? No. Batch has no think time, and will, therefore continually request service from the system. No CPU idle time exists while there is at least one batch job active. How much CPU time batch will get depends on how much is left over after on-line users have been serviced (this assumes that batch is running at lower priority than on-line users). It is conceivable that in our first example batch could be utilizing 45% of CPU resources, and the processor is not at the limit. It is also possible that, in the second case, one user does a matrix inversion. In both mixes response time to on-line users would have been good because of scheduling considerations (there is a scheduling penalty for long transactions). Only if response times are slow and/or batch throughput is minimal can it be concluded that we indeed have a CPU problem. Thus, OPT is most beneficial when used interactively.

EXAMPLES: OPT

The first example illustrates how OPT was used in the following situation: Series 44 with 20 on-line users and one batch job experiencing response time problems. The first step in our analysis is to investigate if response time is restricted by CPU, memory or disc contention, and that system configuration parameters are sufficient for the load. Global resource utilization statistics show that there is no CPU idle time, some memory management activity, moderate disc wait time and a sustained I/O rate of 17 I/O's per second. However, I/O bursts up to 35 I/O's per second are observed at regular intervals. At this point, all three resources are suspect and must be further analyzed. So we enter the User Summary Display to find out if batch gets its share of CPU, and if anybody else is getting an abnormally high share, but all looks well. Next we

focus on memory management activity, and conclude that although memory pressure exists, it is not responsible for the observed response times. Having eliminated CPU and memory, we now move to I/O activity. Disc requests are not evenly distributed over all drives, and for one of the drives lengthy queues exist. We need to identify the reason for the difference between the average and peak I/O rates. Therefore we proceed to the Process Displays. We observe that for the majority of users in the course of a single transaction relatively long waits occur for a lockable resource. From the stackmarker trace we know that we are in IMAGE code during those waits, and the application code segment which calls IMAGE can also be identified. It turns out that we are doing a "DBPUT" to a detail set which is connected to 6 master sets, one being a sorted chain in the detail. The application was optimized by removing the sorted chain, better file placement and database design changes.

Which resource could have been investigated further for possible optimization in this example? Memory usage, since memory pressure existed. Stack utilization statistics could have been explored to determine if stack sizes could be reduced by either a smaller "STACK=" parameter or by use of the "ZSIZE" intrinsic. Working set data could have been used to look at possible improvements in code locality.

The second example illustrates how OPT was used in conjunction with "SOO2101" to troubleshoot a site which was concerned about slow response times on terminals on an MTS (multipoint) line. Please note that "SOO2101"should not be confused with any other user contributed "SOO" named program. It is totally different in content and was developed by a Hewlett-Packard Systems Engineer as an unsupported operational tool to focus on disc I/O and process wait states.

The global data indicated that there was excess CPU time, no memory management and I/O rates bursting to 60 I/O's per second on a Series 44 with a single disc controller. Sequential file processing (i.e. file copying) was suspected because of the achieved I/O rates. OPT didn't show anybody using FCOPY.

The Process Summary Display and individual process displays gave no clues at first glance. A "SHOWQ" command (from within OPT) showed one on-line user on the dispatcher queue over relatively long periods of time, a perfect suspect for someone that is often waiting on disc. Could this user be dominating the disc subsystem

to such an extent that response time would deteriorate for everybody, and why is response only bad for MTS users?

Further analysis of process activity was needed. The data showed that most users coming in over MTS lines were running at a priority of 200, a then known problem in MPE. Since there weren't many non-MTS users, this was not the whole solution to the problem. The I/O bursts needed further attention. OPT does not give a distribution of disc accesses by process. Process states (I/O wait) are the only indicators in this respect, and not particularly useful in sequential file processing because waits rarely occur. So "SO02101" was consulted and confirmed that the user in question was doing about 500 I/O's per transaction. File usage and stackmarker trace data together with information from the application programmer aided in quickly tracing this abnormality to a misunderstanding and logic error when using KSAM.

This last example shows how OPT was used when a new multi-user application was implemented. By coincidence a hardware failure occurred at the same time. Response times fell to 8 seconds. Since the Central Processor was fully utilized without batch, the immediate question was: Why so much CPU in a data entry environment? Process screens gave the first clue: frequent SIR waits. The SIR was identified as the Loader Segment Table SIR and long queues for this resource were observed at regular intervals. Stack marker traces showed that LOADER code was executing on the user stack, as well as FIRMWARESIM code, indicating that the application should be analysed for LOADPROC intrinsic usage, and a the need for a hardware board replacement because the operating system was simulating the decimal instruction set.

Many more examples exist for uses of OPT, too numerous to be detailed in this paper. For instance, profiles for new applications can be obtained and then brought into context with the existing load. Job scheduling and on-line application mix can be tuned to balance resource utilization given that some restrictions are acceptable to the user. OPT is an invaluable tool for troubleshooting as well as tuning. For instance, a program aborted because of insufficient stack space. OPT provided the necessary data to identify that the application had not actually outgrown the available stack space, but was limited by the value assigned to "MAXDATA". A certain level of understanding of how the operating system software works is necessary for interpretation of OPT data.

V. Using APS

As mentioned earlier, APS is an application tuning tool. It provides the basis for WHERE in a program optimization efforts would have the greatest payback by identifying the relative amount of time spent executing various sections of code. Once frequently used code has been pointed out, it can be examined for possible improvements relative to coding, algorithm and task simplification. APS can also be used to identify which programs should be considered for optimization, and for program segmentation.

APS consists of three modules: SAMPLER to initiate measurements; ANALYZER for data reduction and presentation; and DISPLAY for on-line viewing of results during data collection. It can be run interactively or in batch mode. User interaction is facilitated through a series of menus which are in hierarchical order, and prompt for input at each level. No special knowledge is required for running and interpreting application execution profiles. Attention may have to be paid to the sampling interval in some special cases since APS examines software status at regular intervals and then uses this data to extract expected behaviour.

Some preparations are necessary to take advantage of all capabilities offered by APS. During compilation, appropriate compiler options have to be selected to allow mapping of machine addresses to code statements (i.e. VERBS in COBOLII, LOCATION in FORTRAN, LABEL in BASIC, $CODE_OFFSETS ON$ in PASCAL). COBOL and RPG are limited in this respect. Further, during program linkage (PREP), a special parameter (FPMAP) must be specified to include PMAP-like information into the program file.

APS allows you to capture data in the following format:

RM - run and monitor a program
AM - allocate and monitor a shared program(s)
AL - monitor all active programs including MPE.

If "RM" is specified, APS creates the program and the user interacts with the application while sampling takes place. Data collection ends with program termination. For "AM" and "AL" stop times or number of samples have to be supplied. The environment under which a program will be used determines how data collection is set up. For a single user program, new application testing and an initial execution profile for a shared program, "RM" is selected. Shared programs should be further analysed

under multi-user, normal load conditions to reconfirm CPU usage and segmentation ("AM"). "AL" provides a global view of CPU utilization by all user and system programs and processes, thus aiding in nominating candidates for analysis.

APS distinguishes between CPU activity resulting from the execution of user code (Direct CPU Utilization), from system code invoked on a user's behalf (Indirect CPU Utilization), and waits resulting from I/O activity or other blockage. Direct CPU utilization data is needed for code, algorithm and task optimization. Indirect CPU utilization data is valuable for testing an application which is I/O intense, or uses SL segments or Intrinsics heavily. Wait times may be used to identify problems with lockable or other resources and to compute transaction turnaround times by adding all other CPU data.

CPU utilization data is available at three different levels:

1. the program file and process level
2. the segment level
3. at procedure and address levels.

APS reports utilization data in histogram format according to specified parameters. Segment transfer data is presented in a "transition from segment to segment" format.

EXAMPLES: APS

Suppose we have an application written in COBOLII which has 10 code segments. We have taken all necessary steps to be able to do the most detailed program analysis (VERB and FPMAP). We choose "RM" and get the following histogram at the segment level:

Direct CPU Utilization by User Segments:

```
+--------0-----I-----I-----I-----I-----I------%--%CUM-
 %000301 !DD                                    6.5   6.5
 %000302 !DDDDDDDDDDDDDD                       44.0  50.5
 %000304 !DDD                                  10.0  60.5
 %000307 !D                                     3.5  64.0
 %000310 !DDDDDDDDDD                           36.0 100.0
+--------0-----I-----I-----I-----I-----I------%--%CUM-
```

Segment %302 is obviously taking the lion's share of CPU. In addition, segment %310 is also heavily used, and therefore both segments should be considered for further analysis at the next level of detail (procedures and code range).

The CPU utilization breakdown by procedures for segment %302 points to procedure "UTIL1" which should be given further attention.

Procedure UTIL1 - Direct CPU Utilization Over
Code Relative Addresses:

```
+--------------0-----I-----I-----I-----I------%--%CUM-
 %000034 000120!DD                                1.5   1.5
 %000121 000200!DDDDDDDDDDDDDDDD                 19.0  20.5
 %000201 000252!DDD                               3.0  23.5
    .           !
    .           !
    .           !
 %001330 001425!DD                                1.9 100.0
+--------------0-----I-----I-----I-----I------%--%CUM-
```

The data suggests that we should look at the code which generates the instructions ranging from %121 to %200. We consult the VERB map (it has code relative addresses for each VERB) and the source code, and find that we have two statements in a frequently executed PERFORM which do not belong.

We examine segment %310 the same way and find a series of consecutive "IF" statements that look suspicious. Program flow is controlled here by input data. APS suggests that the order in which these tests are done is not optimum. The program is sampled again once the changes are made, this time in a shared environment, to confirm that the changes are yielding the same results during normal load conditions.

CPU usage can be further analysed considering indirect CPU costs. Although the cost of an INTRINSIC (system code executed on a user's behalf) is a given, there may be alternate ways to achieve the same result. INTRINSICS are often designed to handle a task under many different circumstances. Benefits could result from exclusion of generalities, the cost being the efforts expended in writing code. Compilers are also offering various functions which simplify coding or allow mixing of data types in arithmetic expressions, both at the expense of performance. Direct/Indirect CPU histograms will draw attention to these occurrences.

Returning to our example under analysis, there is one more aspect which can be explored: segmentation. Based on the Segment CPU Utilization data, we could consider combining infrequently referenced segments (i.e. %307 and %301). Further guidelines are provided by segment transfer statistics:

Segment Transfer Statistics:

Transitions from user segment		%301 to:
%002	FILESYS1	3
%003	FILESYS4	3
.		
.		
%310		1
Transitions from user segment		%302 to:
%042	CHECKER	127
%043	UTILITY	28
%144	IMAGE01	36
%145	IMAGE02	279
.		
.		
%310		1521
Transitions from user segment		%310 to:
% 171	HIOMDSC1	519
% 202	CLIB'02	5
.		
.		
%302		1318
%310		612

The data suggests that segments %302 and %310 should be combined subject to code segment size restrictions. Frequently used COBOL PERFORM's should be looked at closely to exclude the possibility of a segment transfer within.

APS is a valuable tool for determining program efficiency. APS allows the programmer to review new and existing applications in a production environment with an eye toward application tuning.

RAPID IS A RELATIVE TERM

Jim Dowling

Bose Corporation
Framingham, Massachusetts

ABSTRACT

The features of RAPID/3000 that help to expedite high quality solutions to Information Management problems are described. The trials and success that Bose encountered while documenting thirty odd IMAGE/3000 Data Bases with DICTIONARY/3000. Hints and techniques are provided to simplify the use of TRANSACT/3000 and INFORM/3000. Also, a number of "Extensions" to DICTIONARY/3000 are described which expand the Dictionary towards an Application System Directory.

CONTENTS

INTRODUCTION

"Rapid has been shown to be an effective delay preventive data manipulation tool when used in a conscientiously applied program of good data base design and Religious Data Dictionary management...."

(Anonymous 1983)

The above quotation from a study on the results of three years experience using the RAPID/3000 (and IMACS) products in a business application system development environment has two critical elements; "good data base design" and Religious "Data Dictionary Management". This paper will describe some of the problems that we experienced as we attempted to "LOAD" thirty IMAGE/3000 Data Bases into the DICTIONARY/3000 Data Base. The paper will also present the concept of Closed Loop Documentation. The evolution of this concept involves using the Documentation as the System or in effect; Execution of the Documentation.

Several years ago Bose initiated a search for a "QUERY/3000 Replacement". The objective of this search was to locate a tool for the creation of ad hoc reports from IMAGE/3000 data bases. The target system would provide all of the features of QUERY plus Multi-Data-Set retrieval, a simplified user interface and improved security. We reviewed and/or tested such products as QUIZ, REX, ASK, IQ/3000 and QUERY-B (specifications only). At the time of our evaluation all of these products were quite new if released at all. Our selection criteria led to an ill-fated purchase of IQ/3000. Having lost six months in the search, evaluation and procurement cycle, we were forced to re-evaluate again. At that time we found a new kid on the block, IMACS Corporation was developing the predecessor to the RAPID/3000 products. Essentially we were won over by the concepts of the package namely:

DICTIONARY - A Central repository for the documentation of Applications Systems data and process infrastructure.

INFORM - An incredibly simple ad hoc report process creator that precludes "Change Access" to the data that is being processed.

REPORT - A non-procedural language which provides an upgrade path from ad hoc to production reporting.

TRANSACT - A highly refined data access and manipulation language that removes the drudgery from creating Application Programs.

General - The consistency of presentation to the user and structure of system design that is provided by the TRANSACT processer.

The concept of data access and manipulation being independent of data storage method.

As a result, we purchased all four IMACS products. Our first year's experience with the products was not too productive. There were many bugs in the software and even more problems with our data structures (non-structures). Later versions brought Element Name Aliasing which allowed us to combine a dozen dictionaries into one. Still more revisions and enhancements brought us closer to a viable tool. Just when we thought we were getting there, along came Hewlett Packard and the rules of the game changed again. We were an active participant in the ALPHA test program of RAPID/3000 specializing in DICTIONARY/3000 and TRANSACT/3000. By the time HP had installed dozens of fixes and enhancements we had a good sturdy software tool but very poor documentation. It is not until the current set of documentation was made available that we could even think about turning the package over to our end-users and/or Programming staff. What follows is a capsule view of our experiences with the current RAPID/3000 products and their effect on us.

FEATURES OF RAPID/3000

The heart (and conscience) of the RAPID/3000 product family is DICTIONARY/3000. The Data Dictionary provides a facility for Standardization of Application Systems design and programming that was heretofore unattainable without intensive manual labor. Let's take a look at some programming standards and how the Dictionary provides solutions.

- Report Column Labels

  Stored in the dictionary, one per element.
  Automatically used by INFORM, REPORT and TRANSACT.

- V/3000 Screen Element Labels & Prompt Strings

  Stored in the dictionary, one per element.
  Automatically used by INFORM, REPORT and TRANSACT

- Data Element Print Edit Specifications

  Stored in the dictionary, one per element.
  Automatically used by INFORM, REPORT and TRANSACT.

- Naming of Elements (Variables) in Source Code

  INFORM, REPORT and TRANSACT use the actual IMAGE Data Base Element Names.

  For sequential files they use the Element Names that are configured in the Dictionary.

- Data Element Definition Consistency

  Dictionary/3000 provides a manageable tool for analysts to use to select elements if they already exist rather than re-inventing them for each Application System.

  If Analyst uses this tool many redundancies and incompatibilities can be avoided.

- Recording of Maintenance

  The Dictionary provides an easily secured repository of system data specifications. DICTDBM automatically updates maintenance and create dates.

In addition to Standards enforcement the dictionary provides a powerful repository for program level documentation.

- V/3000 Screens

  The content and descriptive information can be defined and documented in the dictionary.

- Change Authorization Control

  Each entity can be assigned an individual as its custodian thereby indicating a point of coordination for change control.

- Relationships between Procedures

  Programs and Subprograms can be hierarchically related and Where-Used/Explosions can be done to determine change effect.

- Relationships between Data & Procedures

  Data Elements can be related to Procedures providing change effect reporting capabilities.

- Descriptive Text

  Each entity may be described at length within the Dictionary providing an on-line research facility.

INFORM/3000 has many obvious features which make it simple to use but some less obvious aspects are:

- Single Data Set Access at Data Base Level

  This is a disguised blessing. It prevents novice users from choosing an inefficient thread path to connect data sets. Consequently it forces the use of INFORM GROUPS.

- INFORM GROUPS

  This provides a means of documenting what the user's needs are and how the data is being made available.

- INFORM PROCEDURE Directory in Dictionary

  This provides a means of identifying who created what and when. It also assists in monitoring Inquiry usage.

- Read Access Only

  Allows us to provide Reporting from Data Bases and/or files without uncontrolled modification.

- Print Formatting Options

  The ability to modify the output formatting to produce paginated reports or data-only files provides a gateway to other subsystems like DSG/3000 or OPTICALC/3000 without programmer assistance.

- Standardization

  By using standard reporting layouts with Column Headings, data prompts and print editing from the Dictionary is simple to convert an ad hoc report to production level in REPORT/3000.

REPORT/3000 takes advantage of the standardization features that the Dictionary provides. In addition we have found it possible to construct a skeleton structure for REPORT/3000 procedures which enhances the self documenting features and provides a basis for upgrading to TRANSACT/3000. In addition, the following features are noteworthy:

Automatic Generation of Transient Data Elements

Rather than explicitly creating elements for totals, sub-totals, min, max etc. REPORT/3000 does this implicitly for you by simply stating the result that you want.

eg. MAXIMUM (QUANTITY)
TOTAL (AMOUNT)
COUNT (MODEL)

Data Access Thread Control

Tne ability to explicitly establish the path through which REPORT/3000 will acquire the desired data allows the programmer to set up the most efficient path for each report process.

DISPLAY of pre-execution information to the user on $STDLIST helps to ensure that up-to-date documentation is available to the user.

TRANSACT/3000 has two components; the TRANSACT processor and the TRANSACT Programming Language/Compiler. The two work together to provide a programmer with a powerful set of tools for program creation.

THE PROCESSER

- One Program shared by several users.

  With forty users logging on and off through the day and thirty odd simultaneous sessions we can easily see the effects of a :RUN hitting MPE. By sharing code both memory and load time are dramatically reduced.

- Command Structure

  The built in Command/Sub-command display and parser provides a uniform user interface.

- Command Control Options

  User options; REPEAT, PRINT, TPRINT, SORT and FIELD can dramatically increase the flexibility of a System without increasing the complexity of the code.

- Respond Ahead

  An experienced user can respond to several subsequent prompts on a single line thereby reducing wait time and typos due to a sluggish machine.

- Uniform Command Control Codes

The processor handles command block control automatically when the user responds with the codes:

CNTL-Y	-	Processor breaks and returns to Processor Command Interpreter.
RESUME	-	Continue after CONTL-Y
]	-	Terminate current level and return to next higher level.
]]	-	Terminate Command and return to Processor Command Interpreter.
!	-	Auto null respond to all subsequent prompts in an operation sequence.

- Uniform Data Selection Match Criteria

The processor handles the establishment of selection match criteria for data processing by use of a uniform coding and Boolean processing scheme.

Specification Characters

^string	-	Elements ending with "string".
string^	-	Elements beginning with "string".
string1^string2	-	Elements with any character between "string1" and "string2".
string^^	-	Elements with "string" anywhere within it.

Relations

NE	-	Not equal to
LT	-	Less than
LE	-	Less than or equal to
GT	-	Greater than
GE	-	Greater than or equal to

Connectors

TO	-	Establish value bounds.
AND	-	Establish conditional inclusion.
OR	-	Establish conditional exclusion.

TRANSACT PROGRAMMING LANGUAGE

- Dictionary Interface for Data Structures and Presentation

The compiler resolves the storage and display formats for data elements from the Dictionary at compile time. Display edits, prompt strings and column readers are all taken from the Dictionary. This dramatically reduces code generation and improves standardization.

- Dictionary Interface for Data Access Methods

  By resolving data storage methods and access from the Dictionary the differences between IMAGE, KSAM or MPE files is minimized thereby reducing learning time and simplifying upgrades and conversions. By taking care of "Buffer Mapping" the programmer need not concern himself with data storage remapping that may be done to data files to accomodate future requirements.

- Invocation to other TPL programs

  TPL allows a simple "CALL" verb to initiate the execution of another TPL program. Soon REPORT and INFORM procedures will also be available.

- Flow Control Structures

CALL	- Invoke another TPL procedure.
END	- Terminate a sequence, level or program.
GOTO	- UGH!!
IF-THEN-ELSE	- Conditional execution of code.
DO-DOEND	- Establish a code block.
PROC	- Invoke an SL resident procedure.
PERFORM-RETURN	- Execute a block of code.
LEVEL	- Establish a processing level of execution.
WHILE	- Repeat a block while condition is true.

- Dynamic Data (working Storage) Allocation

  By providing a mechanism for control for the amount of data stack that is being used and by requiring that only referenced data need be present on the stack, very large (data) programs can be created without using "Clever" techniques.

- !INCLUDE external Code

  Although the Transact Compiler is very fast it is quite useful to create a large system by building it from modules like ADD, DELETE etc. When each module is complete it can be !INCLUDEd in the end System.

- Simple Debug Methods

  Having the powerful capabilities of the TEST FACILITY and the Comiler Options makes debugging TPL Code simple and effective.

  Compiler Options
  - DEFN - List data item definitions.
  - XREF - List Label definitions and locations.

  TEST FACILITY
  - Invocation - Enter TEST when in command mode.
  - Displays - All registers can be displayed either whenever altered or at specific points of code executions.

- Automatic (Default) Error/Condition Detection and Action

  When a data access or manipulation error is detected or when a data access breakpoint (EOF, empty chain...) is detected, the processor will automatically emit an error, warning or information message then branch to a point in the code block that it determines will provide a point of recovery or continuation.

- Preliminary Data Screening

  When a user provides data to a TPL program, preliminary checks will be made to determine whether or not it is compatible with the restrictions established by its dictionary definition. If a discrepancy is found a message is emitted and the prompt reissued.

CASE STUDIES

THE CASE OF THE CREATIVE CONVERSION

During the conversion of a General Ledger package to HP3000 from IBM (and other mainframes) it came to pass that the KSAM version ran forever and ever. Since IMAGE is so much faster a second conversion was undertaken. The resulting data structure resulted in a most obtuse monument to the programmer's creativity. A sequential file is used to obtain an IMAGE Master Data Set record number. This allows one to sort the sequential file then retrieve from the MASTER SET via directed reads. To compound the problem; there is data in the sequential file that is not stored in the data base (his nifty Binary Search technique is faster than IMAGE could be).

Problem: How do we document this relationship in the Data Dictionary and access data with the RAPID products.

Solution: Create the sequential file in the Dictionary so that we can get at the data. Unfortunately we can not use the Key/Record # structure in INFORM or REPORT procedures but we can "Force" TRANSACT to use it.

THE CASE OF TECHNOLOGICAL SOCIAL CLIMBER

A conversion of an Order Entry System from a microcomputer presented an analyst with an opportunity to reach new heights in his carreer. Data Base Management in the form of IMAGE/3000 was the key to state of the art applications design but "How can I handle this set of multiple record types". Simple; create a Detail Data Set that can handle all the different data elements, make the first element a key to decoding this entry, chain them to an Automatic Master for quick retrieval and simply do not put any data into those fields that do not apply to each transaction type.

Problem: Again now do we explain to the Dictionary that these elements and their locations in an entry are variable and dependent on the first element in the entry.

Solution: Again take heart at least he didn't make it an X(3000) and map it in a Copylib Member only. But the result is still a structure that is nearly impossible to use with INFORM and REPORT.

THE CASE OF THE MISGUIDED OPTIMIZER

A system design required that a transaction log be kept for each Inventory change. This file was to be reported from both on-line and through batch. A Detail Data Set was configured with Part # and Transaction Type chains for efficient on-line access. To save disc space the source of the transaction (eg. PO#, Move Ticket#, Receiver...) was to be stored in a single field called Reference Data. To save on transaction Types; the sign of the quantity field determines the direction of the move (eg. 40 = Issue to floor; -6 means to floor; +6 means from floor).

Problems: Represent the multiple formats of REF-DATA.
Document the arithmetic operations on quantity.

Solution: Again there is not much we can do to help REPORT or INFORM.

THE CASE OF THE DATE WITH A DOZEN NAMES

Across 30 Data Bases we found:

	Name	Storage	Interpretation
1.	DATE	X(6)	YYMMDD
2.	DATE	X(6)	MMDDYY
3.	DATE	P(6)	YYMMDD
4.	DATE	P(6)	MMDDYY
5.	DATE	J(2)	YYDDD
6.	DATE	P(10)	YDDDHHMMSS
	.		
	.		
	.		

Problem: Configure the conflicting storage format and interpretations.

Solution: Assign Aliases to all but the first found.
#6 was a killer on January 1, 1980.

THE CASE OF THE NERVOUS NEOPHYTE

A junior programmer "knocks out a quick system" for a user and decides to use type X fields for all data items. Since COBOL is the language of choice the programmer also avoids problems by using "DISPLAY" formats for all data manipulation.

Problem: How to define this structure so that we can perform arithmetic operations on the numeric data.

Solution: Again no help! The sign overpunch prevents us from using 9, Z, of P formats.

THE CASE OF THE PARANOID PROGRAMMER

This character used passwords at both the Element and Data Set levels but created conflicting specifications.

Problem: Convincing him that he did not accomplish what he thought he had.

Solution: A careful explanation of what the IMAGE/3000 manual states and help in testing his programs with the new structure.

THE CASE OF THE WANDERING DATA

This is the most common problem. The program DICTDBC will generate a schema from the Dictionary Definition of a Data Base. In the process it creates a data element list in alphabetical order (this can not be altered).

Problem: This feature which simplifies schema scanning gives programs that reference data elements by number a real problem with buffer mapping.

Solution: None short of Program rewrite.

THE CASE OF THE NON-IDENTICAL TWIN

We have two data bases with the same structure with the exception of Set Capacities. Each stored in a different group.

Problem: Create two Schemas with different capacities.

Solution: DICTDBC output is edited before DBSCHEMA processing.

THE CASE OF THE CONFLICTING CAPABILITIES

Data Elements in different Bases with the same name have the same passwords assigned with different access capabilities.

Base	Element	Password	Capability
A	Z1	MODIFY	WRITE
B	Z1	MODIFY	READ

Problem: Can not configure this Security.

Solution: Edit DICTDBC output.
Could try setting security at set level.

THE CASE OF THE PASSWORD TO NOWHERE

Two elements in the same data base are assigned to different Security Classes which have the same Password but different access capabilities.

Problem: DICTDBC creates both Passwords in the schema but assigns different Level numbers. When DBOPENed IMAGE uses the level of the first one found. If this is READ then whenever an Access is executed to a set with both elements in it; the second element disappears.

Solution: Edit the DICTDBC output.

THE CASE OF THE ADAPTER/MIS-MANAGER

A quick fix to a Security Oversight in a Data Base using ADAGER works fine but a simple Capacity reduction done months later kills an Application System.

Problem: The change was not incorporated into the Dictionary.

Solution: Centralized/Disciplined control of change.

DICTIONARY EXTENSIONS

or

"When is a small step backward a giant step forward?

There is a program (DICTDBC) that comes with DICTINARY/3000 that extracts the definition of a data base and creates an IMAGE/3000 Root file. On the surface this appears to be a handy time saver but there is much more to it than that. By documenting the structure of an IMAGE/3000 Data base in DICTIONARY/3000 we have created a step towards the development of an application system. This step is the reverse of common practice (ie. get the structure straight then document it). This is my small step backward. The next step in the cycle is to create the ROOT file. This action does not guarantee that the documentation is correct but does make sure that the IMAGE/3000 data base is identical to its documentation. This is my giant step forward. Next we load data into and perform operations on the data base. If changes are necessary, we fix the documentation and Re-gen the data base. This forms a "closed-loop" between the documentation and the data base providing an absolute test of the documentation accuracy. Now if we proceed on with adding Heading, Prompt and Display Edit formatting for each element we create more useful documentation as well as providing more parts of the application system. When we then use INFORM/3000, REPORT/3000 or TRANSACT/3000 to access the data we are in effect "executing" these elements of the documentation.

To document the relationships between programs and subprograms we have established them in the dictionary. DICTIONARY/3000 provides a pretty good facility for creating the Procedures and relating them to each other. This hierarchical representation allows us to locate the usage of subprograms to determining change effectivity. The value of this tool is dependent upon maintenance of the dictionary. We have produced a program that extracts the structure from the dictionary definition and creates an MPE JOB STREAM file which compiles and preps the program. If the documentation is wrong, the program won't work (closed loop). I should note that we store the :PREP options as Description Text since DICTIONARY/3000 does not provide an explicit location for this data.

We recently evaluated two new products from IMACS, ANALYST/3000 and PROGRAMMER/3000. These form a path into and out of DICTIONARY/3000 which comes real close to extending the concept of Executing the Documentation throughout the System Life Cycle. This is an excellent goal for such products.

Here are some additional areas where this concept can be applied:

- Define lists of programs as INFORM groups and create a driver program to present MENUs to users. (See PUDC or PROMAS from contributed Software Library of HPMENU for more detail).

- Create a program to verify the accuracy of V/3000 from files definition against the Form file.

In addition to the above we have also extended the usefulness of DICTIONARY/3000 by using INFORM/3000, REPORT/3000 and TRANSACT/3000 to create supplementary reports and data base maintenance programs. I must warn you not to attempt maintenance other than through DICTDBM without

developing a thorough knowledge of the data structures and interactions that are in DICTIONARY/3000. Some of the reports that we found useful include:

- A list of elements in a data base; DICTIONARY/3000 lists elements in files but no summarized list.
- A list of Data Bases in which a given element exists.
- A complete (Paginated) Element Listing with full descriptions.
- A complete (Paginated) File listing with full descriptions.
- A listing of Orphan Elements.
- A listing of Orphan Files of type MAST, DETL or AUTO.
- A listing of Orphan Procedures of Type SUBP.

We have also created a TRANSACT/3000 program that creates a skeleton program to completely maintain and report on the contents of an IMAGE/3000 Data Base. A current project is to create a TRANSACT/3000 program that will read the compile listing of a TRANSACT/3000 program and document it in the dictionary.

CONCLUSION

RAPID/3000 provides a powerful set of application system documentation and development tools. The capabilities of these tools can best be exploited when "Clean" data structures are used. The effectiveness and quality of the systems created with these tools is dependent on diligent (to the point of dogmatic) Dictionary Maintenance. The potential exists for the evolution of DICTIONARY/3000 into a complete Application Systems Directory. Given all of the above, some imagination and further product development, the term RAPID can be applied relative to most other ways of performing Application Documentation, Programming and Maintenance.

As it stands, with our data structure problem we can do the programming job in COBOL just as fast but it would not be possible to handle the Program Structure reporting, Data Element/File reporting or the Documentation Verification functions. We look to HP, IMACS, ourselves and others to realize the potential that lies in the dictionary and data access tools by enhancement and/or interface products. We came a long way from a QUERY/3000 replacement to a new development methodology. The experience can only be described as Culture Shock and the productivity results are now to be realized. I submit that if RAPID/3000 is approached as a new system it will provide higher productivity than is traditionally seen. If viewed as a COBOL or RPG replacement the results will be limited to the programmer/analysts ability to abandon what he "knows works". In either case the RAPIDity should not be compared to the current programming method but to the time that it would take to do the "complete" job of Documentation, Programming, Maintenance, and Control using current means. I found this to be difficult since we rarely did the "complete" job using cumbersome manual procedures and post facto documentation.

IMAGE LOGGING PERFORMANCE REVISITED

Nestor Dyhdalo
Hewlett-Packard
Technical Support Center
Rolling Meadows, Illinois

With the expanding uses of computer technology and the increasing complexity of data processing applications, the isssues of data integrity and recovery take on increasing importance. The needs of improving data integrity are counterbalanced by the additional costs of data administration, operator involvement and potentially decreased system performance. This paper focuses on the system performance component of data integrity and examines the impact of using IMAGE data base logging under current releases of MPE.

Introduction

When H-P introduced the 1918 MIT (Master Installation Tape) of MPE, they also introduced an enhancement to IMAGE whereby IMAGE would make a copy of the transactions that were posted against a particular data base; this facility was called IMAGE transaction logging. In fact, however, not all transactions are copied (eg. data base reads) but rather only those which make changes to the data base (eg. adds, deletes, updates). In addition, there are certain other transactions which are logged but which do not alter the data in the data base (eg. opening and closing the data base, the use of DBBEGIN and DBEND to define logical transactions); these types of transactions are logged to assist the recovery procedure in identifying which data base a set of transactions came from and also to delineate a set of individual data base transactions as a logical set which should be recovered in its entirety or not recovered at all[1].

IMAGE's transaction logging facility makes use of a logging process which runs under the control of MPE and passes its transactions to the logging process which in turn writes the transaction to a log file either on disc or tape.

In the event that a data base is corrupted (the system will

identify a data base as being corrupted when it detects that there is a data integrity problem which has left the data base in an inconsistent state due to a hardware or software failure), the recommended procedure for recovery if IMAGE logging were used would be to load the last good backup of the data base onto the system and then post the transactions from the log file against that data base thereby bringing the restored backup of the data base up-to-date as well as achieving data integrity. Since this recovery procedure operates in a mode whereby a previously sound (but out-of-date) data base is used and then transactions from the log file are posted against it, this would be a roll-forward type of recovery procedure. A roll-backward recovery procedure (currently unavailable in IMAGE) would consist of starting with the corrupted data base and then removing or backing out those transactions which keep this data base in an inconsistent state.

Implementing the logging facility from a programming standpoint typically requires minimal or even no additional coding above and beyond the calls to the standard IMAGE intrinsics; the only exception would be to add coding to delineate several data base calls (via calls to DBBEGIN and DBEND) as one logical transaction for the logging process for use in the recovery phase. There are of course operational procedures which must be implemented for effectively using logging and most importantly for defining the recovery process.

The Performance Tests

The performance tests were run on a HP 3000 series 64 and series 44 using Heidner's FORTRAN based DBPERF program[2]; they were run in batch mode in the CS queue and all of the logging took place to a single extent disc file. It was assumed that since the hardware on the series 40 is identical to the series 44, similiar results would be obtained for the series 40 with the same configuration.

One can request that DBPERF run with or without son processes which can be used to put an additional load on the system. The type of load (eg. CPU intensive, disc I/O intensive, or memory intensive) would depend upon the type of program launched as the son process under DBPERF. We ran tests that were both CPU intensive and disc I/O intensive as described below.

The data base which DBPERF ran against contained 17 detail data sets; there was a stand-alone data set, a second data set with one path, a third with two paths etc. up to a maximum of 15 paths - for a total of 16 detail data sets; the last data set had one path and this was the set that the son processes used for their update transactions; the DBPERF program ran against each of the first 16 sets; all of the paths were chained back to automatic masters using randomly generated character keys.

In order to attempt to control for configuration differences (eg. number of disc drives, number of GICs, etc.), the tests were conducted using a single 7925 model disc drive under the CIPER release (version C.F0.20) of MPE. The tests were performed on both the series 64 and series 44; the series 64 had eight megabytes of main memory while the series 44 had only two.

Test 1.
The first set of tests were performed on the series 64. Eight son processes which did simple updates to a detail data set were also launched and ran simultaneosly with the DBPERF program. Heidner's DUMYLOAD program was used as the son process. The DBPERF program executed 100 calls to DBOPEN and DBCLOSE, followed by 100 calls to DBPUT and DBDELETE on each of the detail data sets in the data base, and finally 100 calls to DBBEGIN and DBEND. The tests were performed with and without logging enabled.

Test 2.
The second set of tests were identical to those in Test 1 but were conducted on a series 44 machine.

Test 3.
The third set of tests consisted of re-running Test 1 but instead of using the son processes to generate additional data base calls, a CPU intensive program was used instead. Eight son processes were launched which did nothing more than reiteratively calculate the 56th factorial; Heidner's CPULOAD program was used to accomplish this while the the logging test program DBPERF was also running.

The Results

The following tables summarize the data obtained. Each set of 100 transactions is summarized as a mean value

expressed in seconds. The standard deviation statistic is provided to express the amount of variability among the measures obtained. The data for the DBDELETEs are not shown; those transactions took a fraction of a second less time than the DBPUTs but were consistent with the performance data obtained using DBPUTs.

Test 1.
The results for the tests on the series 64 are presented in Table 1. There do not seem to be any significant differences between logging versus not logging on the series 64 with the exception of calls to DBOPEN-DBCLOSE which took an extra 10.7% amount of time with logging enabled than without; the DBOPEN call is an expensive operation for the file system (opening the root file, building the global and local data base control blocks etc.) without even considering the extra disc I/O required by the logging process to be initiated. The extra disc and CPU overhead required for logging did make a difference for this already taxing data base call.

Test 2.
The results for the tests on the series 44 are presented in Table 2. There does seem to be some difference between the logging and non-logging tests, the largest difference appearing once again on the calls to DBOPEN (a difference of 16.1%) and performing 100 DBPUTs to a detail data set with 8 paths took 12% longer with logging.

Since the only configuration differences between the series 64 and the series 44 was the amount of available main memory and the type of CPU in both machines, the data in Tables 1 and 2 present some interesting comparisons between the two machines. Both machines appear to have sufficient memory to run the number of processes needed for the performance tests executed, leaving raw CPU power as the remaining configuration variable to account for the timing differences between the two machines. It appears that the greater the load that is put on the machines, the larger the differences between the two become; performing 100 DBPUTs on a detail data set with many paths (10, 11, 12, and 13 paths) took increasingly longer without logging on the series 44 as opposed to the 64. (See Table 3). The data under the logging condition was not as uniform although the differences were, for the most part, larger than the non-logging condition.

Test 3.
An additional test was run on the series 64 to see whether increases in performance times would be observed in the logging versus non-logging conditions when an additional CPU load was placed on the machine. These results are presented in Table 4 and indeed we now see that the series 64 is beginning to show some signs of tiring as compared to the results in Test 1. The performance data for only four data sets is presented since the CPU bound son processes completed their 2000 iterations of performing the factorial computations by the time the measurements were to be taken on the fifth data set; as a result the performance data for the fifth, and remaining data sets resembled those in Test 1.

Table 1. Resulting means and standard deviations (in seconds) for 100 DBOPEN-DBCLOSE transactions, 100 DBPUTs, and 100 DBBEGIN-DBEND pairs with and without logging. Performed on a series 64 under CIPER with one disc drive.

Type of Transaction	No logging		Logging	
	Mean	Std Dev	Mean	Std Dev
DBOPEN-DBCLOSE	.84	.02	.93	.01
DBPUT - 0 paths	.04	.02	.05	.02
DBPUT - 1 path	.11	.02	.11	.02
DBPUT - 2 paths	.17	.02	.18	.02
DBPUT - 3 paths	.23	.03	.23	.03
DBPUT - 4 paths	.29	.03	.30	.04
DBPUT - 5 paths	.35	.04	.35	.04
DBPUT - 6 paths	.44	.03	.44	.03
DBPUT - 7 paths	.49	.04	.49	.04
DBPUT - 8 paths	.57	.04	.58	.04
DBPUT - 9 paths	.62	.04	.62	.04
DBPUT - 10 paths	.69	.04	.69	.05
DBPUT - 11 paths	.75	.05	.75	.05
DBPUT - 12 paths	.82	.05	.82	.05
DBPUT - 13 paths	.86	.05	.87	.05
DBPUT - 14 paths	.91	.06	.91	.06
DBPUT - 15 paths	.98	.06	.98	.06
DBBEGIN-DBEND	.06	.08	.07	.09
Total Elapsed Time	39.79 min.		40.06 min.	

Table 2. Mean and standard deviation (in seconds) of 100 DBOPEN-DBCLOSE transactions, 100 DBPUTs, and 100 DBBEGIN-DBEND pairs with and without logging. Performed on a series 44 under CIPER with one disc drive.

Type of Transaction	No logging		Logging	
	Mean	Std Dev	Mean	Std Dev
DBOPEN-DBCLOSE	.93	.01	1.08	.04
DBPUT - 0 paths	.07	.02	.07	.02
DBPUT - 1 path	.13	.02	.14	.02
DBPUT - 2 paths	.21	.03	.21	.03
DBPUT - 3 paths	.30	.03	.35	.16
DBPUT - 4 paths	.36	.03	.38	.04
DBPUT - 5 paths	.44	.03	.46	.04
DBPUT - 6 paths	.51	.04	.52	.04
DBPUT - 7 paths	.57	.04	.57	.04
DBPUT - 8 paths	.66	.04	.74	.07
DBPUT - 9 paths	.72	.04	.73	.04
DBPUT - 10 paths	.81	.05	.83	.04
DBPUT - 11 paths	.89	.07	.90	.09
DBPUT - 12 paths	.99	.07	1.06	.09
DBPUT - 13 paths	1.05	.09	1.05	.06
DBPUT - 14 paths	1.06	.07	1.07	.06
DBPUT - 15 paths	1.15	.07	1.16	.06
DBBEGIN-DBEND	.08	.10	.10	.11
Total Elapsed Time	49.57 min.		51.09 min.	

Table 3. Mean differences for the series 64 versus 44 summarized from Tables 1 and 2.

Type of Transaction	No logging Mean Difference	Logging Mean Difference
DBOPEN-DBCLOSE	.09	.15
DBPUT - 0 paths	.03	.02
DBPUT - 1 path	.02	.03
DBPUT - 2 paths	.04	.03
DBPUT - 3 paths	.07	.12
DBPUT - 4 paths	.07	.08
DBPUT - 5 paths	.09	.11
DBPUT - 6 paths	.07	.08
DBPUT - 7 paths	.08	.08
DBPUT - 8 paths	.09	.16
DBPUT - 9 paths	.10	.11
DBPUT - 10 paths	.12	.14
DBPUT - 11 paths	.14	.15
DBPUT - 12 paths	.17	.24
DBPUT - 13 paths	.19	.18
DBPUT - 14 paths	.15	.16
DBPUT - 15 paths	.17	.18
DBBEGIN-DBEND	.02	.03
Total Elapsed Time	9.78 min.	11.03 min.

Table 4. Mean and standard deviation (in seconds) of 100 DBOPEN-DBCLOSE transactions, 100 DBPUTs, and 100 DBBEGIN-DBEND pairs with and without logging while executing 8 CPU bound son processes. Performed on a series 64 under CIPER with one disc drive.

	No logging		Logging	
Type of Transaction	Mean	Std Dev	Mean	Std Dev
DBOPEN-DBCLOSE	5.80	.03	6.40	.03
DBPUT - 1 path	1.41	.67	1.41	.67
DBPUT - 2 paths	1.62	1.26	1.72	1.54
DBPUT - 3 paths	2.74	1.15	2.76	1.79
DBPUT - 4 paths	3.52	1.90	3.58	2.01

Discussion

These data suggest that IMAGE transaction logging under fairly controlled conditions resulted in minimal overhead; on the series 64 the maximum difference was 10.7% on the first DBOPEN issued against a data base while on the series 44 this difference increased to maximum of 16.1%. Additionally, the series 44 showed some further system degradation on the DBPUT transactions. Does this mean that IMAGE logging should be used with every data base? Probably not.

Performance should play only one part, and perhaps a minor part, in determining whether IMAGE logging should be implemented. Certainly if you have high volume transaction rates where recovering transactions manually is impractical then logging should be used. If you need some audit trail (for auditors or some other legal requirements) of the transactions issued against the data base then logging would be appropriate; the audit trail could be provided by a user written routine but IMAGE logging will also capture transactions entered via QUERY. If you have a very large data base (I know of one user whose data base would take three days to DBUNLOAD/DBLOAD), then logging would provide an alternative recovery scheme. Note that performance is not the central issue in deciding whether to use logging or not in these cases.

The tests conducted in this paper, and most other simulations for that matter, are made under ideal conditions; they are not, nor are they meant to be, tests of performance in a specific applications environment. The tests do provide a barometer from which general conclusions can be drawn and also provide valuable information to help make educational inferences about performance in other situations. The tests over simplify the environment within which IMAGE logging will be used. They are over simplified in the sense that a data base application usually does not run stand-alone on a machine; there are usually lots of other users and applications running to complicate the picture. Furthermore, the tests were all executed in batch mode - there was no interaction from a computer user and thus not very realistic; there was, for example, no provision made for typing time nor "think time" - the time spent by the user sitting at a terminal wishing they were somewhere else while contemplating what to enter or not to enter at the terminal, giving the computer time "to think" as well; we

can only guess what he/she/it is thinking. The point of all of this is that application environments are quite unique from a performance standpoint and if performance plays a critical role in the decision to log or not to log, then some performance measurement must be made to identify the impact of logging in the particular environment.

IMAGE transaction logging can also be helpful in the context of obtaining some measurement on the relative performance of an application with and without logging. That is, one can use the logging facility to collect data on the performance characteristics of an application. The resulting log files provide a source of valuable information on the "real-life" performance of the data base in a specific environment. One can measure individual transactions via DBDRIVER but there also exists a utility called LOGLIST by Dennis Heidner[2] which will analyze the log files themselves and produce histograms and other useful statistics.

In the event your performance analyses indicate that enabling logging on your data base applications causes significant degradation on your system and you are concerned about data integrity, you might consider performing the data base adds and deletes in batch mode during off hours. IMAGE has been optimized for reads (DBGET) and updates (DBUPDATE). These calls typically do not require as much system resources as adds and deletes (DBPUT and DBDELETE). Any data base adds could be posted to transaction files and then batch posted to the data base during low activity periods. The data base deletions could be implemented by creating a delete flag item in the particular data set and then performing the actual deletions in batch at a later time. When data base failures or errors (broken chains, incorrect chain count etc.) occur, they are usually preceded by some transaction which alters the structural integrity of the data base (ie. via adds and deletes). These types of data base integrity problems could thus be minimized by performing the add and delete transactions during low activity levels; a backup of the data base should be made before during the batch postings and then one immediately after. Should some error occur during the batch posting, the backup could be restored and the posting jobs run again.

References

[1]Hewlett-Packard. IMAGE Data Base Management System Reference Manual. Sept. 1979 (2nd Edition).

[2]Heidner, Dennis. Transaction Logging and It's Uses. 1982 HPIUG San Antonio Conference proceedings.

ARE COMPUTER GRAPHICS JUST ANOTHER PRETTY FACE?

Tim Dudley
Quasar Systems Ltd.

We are facing an interesting situation in the evolution of computer graphics. After years of struggle, a sense of elitism among computer graphics system designers and users, smugness among graphics gurus, and frustration in people who have had to fund these systems, we are now at a point in time where computer graphics has finally come of age. Computer-aided design systems abound for both mechanical and electrical design, solids modelling systems are becoming more widely available, systems in general are becoming cheaper and more sophisticated, and the phenomenon of "business graphics" or "management graphics" has arisen. The concensus of market projections indicates that the business graphics business will increase at a rate of 40% per year, and is in fact the fastest growing segment of the computer graphics market. But when examined more closely, it turns out that there isn't much agreement on what business/management graphics is. In spite of this, there are at least sixty companies in North America who count themselves as suppliers of business graphics.

What is business/management graphics? There's more to it than bar charts, pie charts, and histograms. But it's not clear that the majority of suppliers of so-called business graphics systems really understand the problem of graphics for management. The emphasis has been on the production of high quality pictures - presentation graphics - with a multitude of colors and attractive layouts. There has been too much attention paid to cosmetics, and not enough to content. The user interface has generally been badly done, or not made specific enough (it doesn't need to be ideosyncratic, but needs to be something more than user-friendly); this is probably because the User hasn't been identified. And presentation graphics doesn't really address the central problem - that of garnering information from a management database in order to facilitate decision making. The power of the computer needs to be made available to the decision makers directly, and not filtered through a graphics department, and then a graphic arts department. The classic Harvard Business School description of a manager's primary

function in an organization is the assimilation of information and the production of decisions. A significantly large proportion of the data containing the information on which those decisions are based often resides in a corporate database. It should be the purpose of a business graphics system to directly support that function by making the pertinent information easily accessible, thereby facilitating the high-level decision making process.

So, has computer graphics really come of age? I don't think so. There are significant shortcomings in almost all of the currently available business graphics systems, which together seem to indicate an erroneous approach to the problem of supporting the managerial function. And this has most likely resulted either from a misunderstanding of the use of graphical representation of information, or from a laziness on the part of some system suppliers; they appear to have jumped on the bandwagon, solved a small subset of the entire problem, and attempted to capitalize on display technology (both hardware and software) which may not be suitable for providing management information to the people who need it, when they need it. And the people who need the information shouldn't have to think like the computer in order to use it.

While computer graphics has been around for just over twenty years, graphics for management is relatively recent, the boom occurring only in the last three or four years. This boom is largely attributable to the technological advancement in raster graphics displays, and the precipitous drop in hardware costs. These two factors have made graphics display systems relatively inexpensive, and have moved them within reach of organizations other than large manufacturing companies, research organizations, and the military-industrial complex. During this process, graphics system designers began to appreciate the problems of the user interface, and did something about it. At the same time, it was recognized that a potentially huge market existed for business graphics, including pictures from forecasting models, showing trends, exceptions to trends, and interrelationships among data. And the actual images required to represent this information were comparatively simple, particularly when considered with some of the CAD work being done in both the mechanical design and electronic design areas. The display technology had advanced to the point where very realistic pictures could be produced, and designers were beginning to understand how to manipulate and structure data in order to support the rapid display of very complex pictures. All of this technology (again, both hardware and software) was available, and a huge untapped market was identified. For

the most part, access to this market merely meant the addition of some application software, or a programmable interface to an already existing graphics package, and POOF! You were in business! Because the systems could produce very high quality pictures, this aspect was, and still is, used as the major selling point for these systems.

As already mentioned, the actual images making up a graphic for management tend not to be very complex, consisting largely of bar charts, pie charts, histograms, scatterplots, and X-Y line plots. All of these are easily generated with a minimal set of graphics primitives such as circle, rectangle, polyline, polymarker, fill, and some text routines. Business graphics images were a small subset of the images which could easily be displayed with most graphics systems, and could be generated without too much effort. Graphics editors existed and were well understood, as were the techniques of manipulating display files, so the next most popular selling point of these systems was usually the ease with which a graphic could be modified. A recent ad by an unnamed manufacturer illustrates this. The ad shows a plot of a pie chart with a memo clipped to it, saying something like "...Jim - sorry about the rush, Fred's leaving for San Diego at 4:30 and needs the indicated changes made to this chart before he goes...thanks " , and the memo was written at 2:00. But, thanks to the unnamed manufacturer's super business graphics package, Fred makes his plane to San Diego with a very classy version of the same plot. No new information has been added, but the second version looks much better than the first. This is a manifestation of what is sometimes referred to as the "Proctor and Gamble Syndrome" - that is, people generally do not associate quality with a lesser image. Conversely, (and this may be the core of the problem) they do associate quality and validity with a high quality image. This tendency results in even more emphasis being placed by the system suppliers on cosmetics, rather than on content.

The other situation which has developed, and which is alluded to in the above-mentioned ad, is that the systems are generally not amenable to use directly by the people who require the information from them. Because of the emphasis on presentation quality, the systems are generally directed toward a user who will produce a graphic for use by someone else, most likely for a presentation. This approach introduces a middleman between information and the person requiring the information. The middleman is typically a programmer, a clerk or administrative staff person, or a lower level manager (who may also pass the task off to a clerk or programmer, thereby removing the information one level further from the person requiring it). The introduction of this

middleman puts a certain amount of guesswork into the selection of the required information. It is still being filtered through a form of graphics department and then a graphic arts department; the difference is, that it just doesn't take as long as it used to.

This is not to say that presentation graphics systems are not useful. It has been demonstrated and documented by numerous studies that people retain information longer when it has been presented graphically than when it has been presented in other ways. Meetings are dramatically shortened by use of graphics in presentations, and the so-called information float (the time between the availability of information and its use) has been significantly reduced by the use of computer-generated presentation graphics. The reason for this is that computer graphics systems turn data into information. The much heralded information overload in an organization has been misnamed. It is in fact a data overload. This is an extremely important distinction: data is the raw material from which information is obtained; information is the knowledge or meaning extracted from the data. According to David Friend, chairman of the board of Computer Pictures Corporation, the mere existence of data doesn't guarantee their usefulness, and more data can actually yield less information. Presentation graphics addresses the problem of extracting information from data, but expends too much effort in making the presentation look good, as opposed to making more information accessible. The need is to show information, in a way that leads to better management decisions.

It must be remembered that it is the decisions that are the end product, and not the presentation of the information on which those decisions are based. What we have is a solution looking for a problem: presentation graphics systems which have been attached to a user interface, and sometimes (infrequently) also interfaced to a graphically oriented database. What we need are management information systems which are capable of producing graphical reports as well as textual reports. The graphics should be treated as an output stage of a total information system, and not as an end in itself. We need systems which are capable of more than the production of advertizing quality pictures - we need systems which enable decision makers to explore their databases, look for trends and exceptions, examine interrelationships among data and sets of data, and to do this quickly and easily. The entire motivation behind management graphics should be to support the decision making process, and the interaction with the machine should aid that thought process, not impede it. High quality, full color, high resolution graphics are neither necessary nor sufficient for these kinds of systems.

Several areas must be addressed before these systems can be successfully produced. One of these is the problem of the perceptual and cognitive approaches to information. Management information, by its nature, can be considered to be layered, with the outer layer consisting of perceptual information, the next inner layer consisting of cognitive information, and the innermost layer consisting of the data which contains the information. Perceptual information tends to be highly loaded - lots of information, but not much detail. Questions based on this level of information tend to be answerable by "yes" or "no", such as "Is there a trend developing", "Are there exceptions to the trend?", "Does this entity intrude on that one?". Cognitive information is more precise, contains more detail, and tends to be measurable.

The difference between these two levels of information is best illustrated by an example. A few years ago, the Department of Indian Affairs decided to put a road into one of the federal parks, which would give tourists a good view of the park. There were some eyesores, such as a garbage dump and a water tower, which needed to be hidden from view, and a lake which was located some distance from the corridor which the road was to penetrate, which the Department wanted to be visible as part of "a good view". A graphics system was proposed which displayed a black and white contour map of the area, with pertinent features identified. The landscape architect would trace a proposed right-of-way for the road onto the contour map, then rotate the map 90 degrees about the horizontal axis, define the height of an observer's viewpoint above the right-of-way, and then do a simulated flyover of the contour map, following the right-of-way. The architect would have the capability of stopping the flyover at any point, and of directing the observer's line of sight by rotating the map about the vertical axis. Using this technique, it would be possible to determine which features were visible and which were hidden at any point on the proposed right-of-way. The images involved in this phase of the application were very sparse, and relatively abstract, but contained sufficient information to enable the architect to explore several alternative rights-of-way very quickly, and to disregard the ones which violated the first level of acceptability criteria. All of this would be done using perceptual level information.

Once the architect had selected two or three viable alternatives, based on the perceptual level acceptance of the visual criteria, then significantly more complex programs would be utilized, imposing gradient constaints and turn radius constraints on the roadway, performing cut and fill calculations, and so forth. These calculations would be done at the cognitive level. The final images at

this level would include much more detail, and be less abstract and more realistic.

The point here is that the process was deliberately divided into two distinct phases, based on the level of information required for each phase. By working first at the perceptual level, several alternatives could be explored, prior to committing more significant time, manpower, and monetary resources to the project, and subsequent effort could be directed toward exploiting viable alternatives, thereby providing a better solution to the problem.

The same principle should be applied to management graphics systems. Again, from David Friend: "80% of management decisions are made with 20% of the information. The key to a successful management graphics system is in providing managers with an instant look at the core 20%." Also, in many cases, a rough cut at the numbers as soon as they are available is worth a lot more than an audited accounting a month later. The analogy extends to the printed page. In most cases of information transfer within an organization, the information is exchanged either verbally or on a handwritten page. Decision makers are more interested in what the information is, rather than how it looks. They are more interested in content than cosmetics. The same principle should apply to computer generated graphics: publication quality output simply isn't required in order to convey information.

Another area which still needs attention is the user interface - the system must be designed with the user in mind. In the case of management graphics systems, potential users fall generally into four classes: executives, managers, programmers, and administrative staff. Executives and managers don't want to create graphs, they want to see them. With presentation graphics systems, the emphasis is on the ease of creation and editing. But this begs the question. Systems should be consciously split between the graph creation portion, and the graph storage and retrieval portion, possibly with a different user interface for each portion (although this approach still encourages the use of a middleman to store the graphs he or she thinks the decision maker will require).

A better approach is to ensure that the user interface is designed to be convivial with the intended user of the information - the decision maker. That interface doesn't necessarily have to be ideosyncratic, but it does need to be more than just user-friendly. Again, the user shouldn't have to think like the computer in order to use it. The system must aid the thought process of the user

in order to be useful. This means combining a good user interface with a perceptual approach to the information in the database, giving the decision maker the capability to browse the database, identify trends and exceptions, look at relationships among data, and do this at the perceptual level first. This provides rough cut, snapshot views of the information database, with not much detail, but with very high levels of information content. The graphics portion of the management information system should then be treated only as an alternative information delivery system: a graphical report generator. The user interface to this type of system is critical, in that it must not get in the user's way. In addition, the user must be clearly identified early in the system design process, so that the function of the system can determine the form of the interface, and not vice versa.

A third area which must be addressed is that of database manipulation. As mentioned earlier, the majority of the so-called management/business graphics systems are solutions looking for a problem (there are a few notable exceptions). They typically provide a menu for the user to - ready? - KEY IN the data values to be graphed. The systems then dress up these numbers and make pretty pictures out of them. Some systems also provide programmable hooks to an application database which allow a graphics programmer to read the application database in order to create a graphically oriented subset of that database, which can then be used to produce presentation quality graphics. Both of these approaches encourage the use of a middleman, and in the case where the numbers must be keyed in, the production of graphics becomes an extra step in the information transfer process, and not an integral step, as it should be. Neither of these approaches provides a satisfactory solution to the problem of manipulating a database to make the information contained in it readily available.

There is no longer any question that information which is displayed graphically can improve both the quality of managerial decisions, and the decision making process itself. However, by concentrating the emphasis on high quality images, half of the medium's potential is being ignored. In order to more fully exploit the medium's inherent effectiveness in making information available to a user, systems need to be designed with the following considerations:

i. The distinction between perceptual and cognitive levels of information must be consciously recognized, and both levels supported by the system.

ii. The functional distinction must be made between the creation of graphs, and the storage and retrieval of graphs, and both of these functions supported.

iii. The functional distinction must also be made between the highly interactive graphical browsing capability, and the much less interactive presentation graphics capability, and both of these functions supported. This is one of the major weaknesses in systems which are currently available: the browsing capability is largely nonexistent. Ideally, the system should provide both graphical and textual browsing capabilities, with the facility in both cases to add more detail to the reports as required, and to then produce presentation quality material with supporting textual reports. These capabilities should all exist in the same system.

iv. The graphics portion of a system should be treated as an output stage of a total information system, and not as the end in itself. Textual report generators exist. What is required is an addition to the information delivery system, in the form of graphical report generators.

v. Graphical reporting must be closely tied to powerful database manipulation capabilities. It must be easy for the user to get at information, as well as to graphically report it.

vi. The property of "conviviality" is the utility of a tool as perceived by ordinary people. For a thing to be convivial, it must be easy to use, and not require significant levels of training before there is recognizable utility in its use. In order for a system to be successful, the intended user(s) must be identified early in the design process, and the user interface made convivial for those users. The user interface on many management graphics systems preclude managers from directly using the system. It shouldn't.

If systems are to be designed with users in mind, we must know who those users are, what their functions are, and then design systems to support those functions. The function of a manager is to assimilate information and to produce decisions. Management graphics systems can have a very beneficial impact on that function. But it is time to put the power of that medium directly into the hands of the decision makers. Computer graphics are not the end in themselves; they are the means to an end. And that end is better management decisions.

Acknowledgements:

Much of this material was motivated by ideas from the following people:

i. David Friend, chairman of the board of Computer Pictures Corporation, and founder of Friend Information Systems, Boston, Massachusetts, USA

ii. George Roy, artist and system designer, Perle Systems, Toronto, Ontario, Canada

iii. Gordon Thompson, futurist and self-described court jester, Bell-Northern Research, Ottawa, Ontario, Canada

MPE I/O System Overview or Where Do You Go After LDEVS?

Rick Ehrhart

Hewlett Packard

Introduction

This paper will give a brief overview of the MPE I/O seminar. The seminar will cover the data structures, the procedures, and the hardware of the MPE I/O system. The data structures include the Device Reference Table, I/O Queue Entries, and the Device Information Table. The I/O procedures will include ATTACHIO, SIODM, and I/O drivers. The I/O hardware will be shown with a brief discussion on channel programs.

Data Structures

I/O tables are need for three reasons: 1) to enable the software to find the I/O driver, 2) to enable the hardware to find the I/O driver, and 3) for the I/O system to keep track of the I/O in progress.

The Device Reference Table (DRT) gives the hardware access to the I/O system. It provides the hardware with a plabel to the interrupt procedure and a pointer to the Interrupt Linkage Table (ILT) which points the driver's Device Information Table (DIT).

The Logical Physical Device Table (LPDT) gives the software access to the DIT. This table maps the Logical Device number (LDEV) to the driver's DIT.

The Device Information Table (DIT) contains the needed information for the driver to keep track of the I/O in progress. It points to the ILT which contains the channel programs for the device.

I/O Procedures

The I/O procedures control the queueing of I/Os, and the state of I/O in progress. The procedure ATTACHIO queues an I/O onto the driver's DIT. The procedure SIODM manages the state of the driver. The driver builds the channel program, starts the execution of the channel program, and cleans-up after the channel program is finished.

Hardware

The I/O hardware for the 3000 IMB based systems consists of a General I/O Channel (GIC), and the Channel Program Processer (CPP). The CPP executes the channel programs that instruct the GIC what HP-IB lines to toggle. The GIC is one board that can have eight devices attached to it, like discs and line printers. The terminals attach to the system via the Asynchronous Data Communications Channel (ADCC). Four terminals can attach to it.

HP Business BASIC: The Early Years

Dave Elliott, Project Manager
Hewlett-Packard Company
Information Networks Division
Cupertino, California

Abstract

With the BASIC language supported on as many as fourteen different computing products within HP, the need has developed for a standard definition of a commercial BASIC which will allow the migration of applications between various machines. Using the highly regarded BASIC/250 as a starting point, the languages section within IND has defined 'HP Business BASIC' which is now being implemented on several processors including the HP3000. This new BASIC will contain embedded interfaces to application tools such as database management, data entry forms and a report writer. Presented are the history and objectives of this effort. Trade-offs are discussed between functionality, compatibility and friendliness when defining a language to be used in widely varying programming environments and among users of widely varying sophistication.

1. Introduction

Language implementers seldom get the opportunity to be language designers. Constraints imposed by industry standards and compatibility objectives usually prevent the development of new language features or the modification of existing ones.

The HP Business BASIC project, which has had very strict compatibility objectives, would seem an unlikely area in which any creativity could be exercised. However, the fact that the project is aimed for implementation on several different machines has necessitated the removal of system-dependent syntax and the development of generalized language constructs which can be meaningful in different system environments.

2. History

The HP Business BASIC project was started in the summer of 1981 by joining together the BASIC projects from Information Systems Division (now Information Networks Division) and General Systems Division (now Personal Office Computer Division). The ISD group consisted of several engineers (including myself) who had had a great deal of experience using and developing software for the HP3000. The GSD group consisted of engineers who had used and developed software for the HP250. The merging of the groups resulted, at first, in a fierce philosophical battle about the nature of the BASIC language.

Note: The specifications in this paper are supplied for informational purposes and may be subject to change. Providing this information does not constitute a guarantee, implied or expressed, as to the features or availability of HP Business BASIC.

a. The HP3000 View

BASIC is one of a set of language subsystems offered on the HP3000 system. It has an advantage over COBOL, Fortran, Pascal and others in that it has an interpreter. BASIC users have access to certain tools which also reside on the 3000 as subsystems. If a BASIC user wants to have an Image database in an application, for example, then a calling mechanism is provided in BASIC to access the Image subsystem. A knowledge (i.e., manuals and/or training) in Image/3000 is required. The BASIC subsystem is under the control of a complex operating system (MPE) and users can interface with the MPE file system or use BASIC data files to store information.

b. The HP250 View

BASIC is the means by which users access all of the functionality of the HP250 system. There is no concept of subsystems. There are no other languages, nor is there anything to 'call' other than user-defined subprograms. The 250 user has a complete program development environment using BASIC as the command language. *Database manipulation, screen management, sorting and report writing are language features, not outside services offered under the umbrella of an operating system.* On the 250, there is no conceptual difference between the language, the operating system or the application tools. The HP250 has a very loyal user community and has consistently scored very highly in surveys of customer satisfaction. This is due to the ease of generating sophisticated applications totally within BASIC's interactive user environment.

The first few project meetings were stormy affairs. The group had the charter of implementing a BASIC on the 3000 which would allow the migration of HP250 applications. It was not clear to the project members that such an effort was feasible. The point was made that the only major thing that BASIC/250 and BASIC/3000 had in common was that they were both called BASIC. In terms of the functionality offered and in the role of the language in the overall system, the two BASICs were poles apart.

An early discussion centered around the necessity of duplicating the BASIC/250 language syntax for database, screens and report generation. The ISD (HP3000) contingent felt that such functionality was best provided by allowing calls to Image and Vplus. They felt it was not the language's responsibility to, in effect, replace the user interface of the various application tools with syntactic extensions. In the 3000 view, this would be inconsistent with the operation of other language subsystems. The 250 group was adamant that not implementing these statements was tantamount to not implementing BASIC/250 at all. They pointed out that only the Image statements within BASIC/250 could be directly translated to calls to subsystems. The Forms (Vplus) capabilities on the 250 were embedded in the *semantics* of the PRINT and INPUT statements. The 250 Report Writer had no counterpart on the 3000 and would have to be implemented within the new BASIC if there was to be any hope of migrating 250 applications.

It became apparent that the only way to determine the best approach was to examine our constituency (current and future users of BASIC on HP machines) and develop a set of objectives which addressed our charter and would result in the most benefit to all members of that constituency. We realized that the HP250 customer was in great need of a BASIC running on large machines which was highly (if not totally) compatible with BASIC/250. The 250 has a limit of six users (currently being expanded to ten). Applications written on the 250 have no way of migrating to other systems. There is also a need to bring the 250 into the 'family' of HP business computers, allowing new users to start with a 250 application and grow up through the product line. The BASIC/3000 user, on the other hand, has, and will continue to have, a BASIC interpreter and compiler which run on the largest machines that

HP offers. The feature set of BASIC/3000 is such that almost any new BASIC will provide much more functionality and user-friendliness. The compatibility aspects of a new BASIC are not critical for BASIC/3000 applications. New programs can be written in HP Business BASIC and old programs can continue to be run in BASIC/3000

3. Objectives

The following objectives evolved over a investigation period lasting five months. During this time, the differences between the ISD and GSD contingents of the project began to dissipate. Each group made a serious effort to understand the needs and problems of the other. The project, which had originally been in the GSD organization, moved to IND early in 1982 and began its Specification Phase with the following objectives.

Develop BASIC interpreters and compilers which run on the HP3000 and future products throughout HP's business computer family.

In the past, language processors were developed for specific target machines. Code generation was a major portion of a compiler development effort and this, of course, was machine-dependent. However, because HP systems use several different processors (3000, 9000, Motorola 68000, Z-80, etc.), a common intermediate language has been developed which will become the target 'machine' for most new compilers. Code generators have been written to translate this intermediate language into several different machine languages. This makes our first objective feasible. The new BASIC is being written in Pascal so the system itself will be transportable between machines.

Develop a set of products which are friendly, easy-to-use and are designed to be of high quality.

Although this sounds like a 'motherhood and apple pie' objective, it is one that is taken very seriously by the project team. It is clear that products which fall short in this regard cost the company a great deal in terms of maintenance resources, support costs and, most importantly, customer satisfaction.

Provide a complete application development environment to include interfaces for database management, data entry screens and report generation.

The programming environment of the HP250 was to be our model for the programming environment of HP Business BASIC. The HP250 users to whom we spoke were convinced that their very high productivity was the direct result of the integration of application tools into the programming language. The debugging features of BASIC/250 are very powerful and include single-stepping and the ability to modify running programs.

Provide at least 90% compatibility with current HP250 applications.

We defined 90% compatibility as meaning that the effort required to convert an application would be approximately 10% of the effort required to rewrite the application in another language. This, of course, is a difficult objective to measure but the adopted definition was considered more informative to the potential user than simply stating that a certain percentage of lines, statements and/or programs were compatible. Conversion tools would be provided for BASIC/3000 and BASIC/250 applications which would perform the translation to the new syntax.

4. Design Considerations in HP Business BASIC

Following are several examples of BASIC/250 features which have had their syntax or semantics changed for HP Business BASIC. The purpose of these changes was to make the features more machine-independent and to improve the consistency and friendliness of the language. Each of these changes can be handled by an automatic translation in the conversion process.

a. The PRINTER IS Statement

The HP250 programmer can direct the output from the PRINT statement to various devices using the PRINTER IS statement. The syntax is:

PRINTER IS <numeric expression>

At run-time, the numeric expression is evaluated and rounded to the nearest integer. Subsequent PRINT statements send their output to the device corresponding to that integer in a table specified at system configuration time. "PRINTER IS 8" usually means that output is directed toward the terminal screen. "PRINTER IS 9" usually specifies that output goes to a 'bit-bucket' ($NULL on the HP3000).

It was felt by the project team that this syntax was inappropriate for the HP3000 and imposed upon the user the burden of remembering numbers for various output devices. The first decision we made was to allow keyword device specifiers. Thus, "PRINTER IS DISPLAY" would replace "PRINTER IS 8" and so on. However, when specifying that output should go to the system printer (device 10 on the HP250), the somewhat awkward "PRINTER IS PRINTER" resulted. We therefore changed the syntax again to the following:

[SEND] OUTPUT TO <printer-spec>
<printer-spec> = DISPLAY / NULL / PRINTER / PLOTTER /
<string-expression >

On the HP3000, if a string expression is used, it will be evaluated as an MPE file name (or back reference to a file equation). The file may be an output device, an MPE ASCII file or a BASIC data file.

b. The Image Statements

The interface to the Image database subsystem in BASIC/3000 consists of a number of predefined subprograms which convert their parameters and then call Image/3000 intrinsics. These subprogram calls look very much like intrinsic calls (i.e. unfriendly). Surprisingly, the Image statements on the HP250 *also* look like intrinsic calls although the keyword CALL has been removed. BASIC/250 was apparently designed this way *to be 'compatible' with BASIC/3000, even though there are no Image 'intrinsics' on the 250.* We decided, for ease of use and consistency, to develop a set of true Image statements in HP Business BASIC that would be self-documenting and would minimize the work of formulating the parameters for intrinsic calls. By combining keyword and positional parameters, the Image statements (DBOPEN, DBGET, DBPUT, DBUPDATE, DBDELETE, DBFIND, DBLOCK, DBCLOSE, etc.) have been made much more readable and self-documenting. To demonstrate this, the following are examples of a DBGET in the two existing BASICs and in HP Business BASIC.

```
BASIC/3000
   CALL XDBGET(N$,"Bolts",3,S(*),"@;",P$,Q,P2)

BASIC/250
   DBGET("Parts","Bolts",2,Error(*),"@;",Buf$,0)

HP Business BASIC
   DBGET "Parts" INTO Buf$, DATASET="Bolts", MODE=SERIAL, &
      STATUS=Error, ITEMS="Part_name$,Quantity,Price"
```

c. The MASS STORAGE IS (MSI) Statement

Many BASIC/250 programs start with a statement such as:

```
100  MSI ":C2,7"
```

This is a directive to BASIC that all files referenced (until the next executed MSI statement) are to be found on a particular disc drive. Files with the same name may reside on different mass storage devices and this statement is used to differentiate them. The file system on the HP3000 would not easily support such a feature, nor is it apparent that such a feature is particularly necessary or desirable on the 3000. We could hardly translate the statement above into MSI IS "LDEV3". Instead, we realized that the MSI statement can really be thought of as a set of file equations. It equates all filenames occurring in the program to files residing on a certain disc or volume. In converting BASIC/250 applications, we would be moving HP250 files from volumes into groups and accounts. Thus

```
MSI "grpname.acctname"
```

would be the appropriate syntax in HP Business BASIC. This, of course, is somewhat obscure. To be more readable, the syntax was changed to

```
FILES ARE IN "grpname.acctname"
```

This, in effect, would be the same as the (mythical) file equation

```
:FILE @ = @.grpname.acctname
```

5. Conclusion

The specification of HP Business BASIC has given the project team several opportunities to design new features and modify existing ones in BASIC/250 and BASIC/3000. All decisions made during this process are done within the framework of our project objectives. The HP Business BASIC language is scheduled for implementation on several different machines and will be an important HP product for the next ten to fifteen years. The project team must make sure that the language is appropriate for a variety of users, ranging from the experienced programming professional to the six-year-old learning to use a personal computer. Working with the HP250 as a model, we are very confident of success.

LEST WE FORGET...THE END USER

Tom A. Elliott
Commonwealth Theatres, Inc.

There is a constant fight being waged between the two major factors of a given "system"...the End User and Data Processing.

This presentation is designed to answer a very simple question:

"IS THE FIGHT REALLY NECESSARY?"

In order to make such a determination we need to investigate the various people involved and their thought processes. What you will read in the following pages is nothing you really don't already know. The only thing that may make this paper different is that you may not have been confronted with reality in black and white. The value of this process will be discussed at the end. (NO FAIR PEEKING!)

The most valuable asset of a company is it's people. Everyone has a niche, and, as long as they carry out their responsibilities, the company will have a chance to not only survive, but prosper. Given that the data processing staff and the computer hardware are generally not revenue makers and that they consist of a very large cash outlay, they provide the rest of the company with an opportunity to take pot shots at the mystery black box. Who can argue with them?

We'll be doing a little tongue-in-cheek fun-poking at the various people within the company, knowing that a sense of humor is not only of interest to all of us, but is mandatory for survival and success in the mystique of computer processing. It may, indeed, be our salvation.

No matter what business you're in the impact of the computer revolution can be seen daily. Check-out stands at the grocery store have shortened the amount of time you have to stand in line, credit card checks (Have much do I have left on that card?), banking with a machine that dispenses real honest-to-goodness money, and calling to check on you mail order (Sorry, the computer is down and will not be back up until 3:30 a.m. Please call back then...this is a recording).

The basic responsibility of the data processing department is to provide services to all other departments, with the ultimate goal of providing better information faster and more accurately than can be done manually. Without the end-user to service there would be no need for data processing. Lest we forget, the end-user is the heart of the company.

DEFINITIONS

The following glossary is not another attempt to push thoughts on how you should define complex issues in the field of data processing but is rather a beginning point so that you can follow my thoughts throughout this presentation.

END USER: Any non-data processing oriented person that actually uses an application program.

DATA PROCESSING: A person (or department) that created and/or is responsible for said applications program.

ERGONOMICS: A fancy term whose major thrust is to make the end-user forget they are using a computer; instead they are consulting with a friend. (Would you buy a used car from this friend?).

MANUAL: This type of a system represents one accomplished with the pencil and the paper, completely without the help of automation (overlooking the calculator, of course). For many end-users, this is the best system designed for there use...and for some end-users they are right.

BATCH: When a lot of data is sent to the computer across a varying time period, the computer segregating it into smaller units. We think of these small units as batches. Typically the computer is given batch balances with which it can make sure all of the data in the batch has been received and entered before it goes on. The key word in this process is TIME. There may be relatively long periods of time that will elapse before a batch can initially enter the computer, be validated for errors, and the errors be corrected until they are actually stored as data for future processing.

ON LINE: Basically a simple term, it describes the end-user being able to converse directly with the computer, getting information as it exists at that instant.

REAL TIME: Although dependent upon being on-line, it describes the ability of the end-user to enter data directly, have it validated, correct it, and store it immediately.

YOUR BASIC END-USERS

Everyone in data processing knows that there are different kinds of end-users. They can be classified quite easily. By knowing the type of end-user you are working with you can generally find a way to handle them.

For the sake of simplicity, we will give you a smattering of these definitions. We're sure you know of others and can add to this list ad infinitum.

THE BEAST This person is most dangerous when they are injured. Prove they were wrong and they will start dragging all of the skeletons out of your closet. Such things as "Response time has really been bad lately." and "I'm not getting any reports on time. How come?". Little half-truths are gently dropped to your superior right after you leave the room. "Have you noticed that all projects seem to be taking more time? I'm sure that the fact that the d.p. staff doesn't start work until 9:30 and are the first to leave has nothing to do with it!. Anyway, it's probably just my imagination."

THE DOUBTER This individual is so sure the system will not work that they can be self-prophets. Before the system goes "live", they are confident they can do it better and faster manually. Typically, when it does go "live", they remember a series of exceptions that no one has taken into consideration during the design stage. The negative attitude affects many phases, including response time. Three second response for the doubter is a long time. The same three seconds for a user with a positive attitude will be just great.

THE INTELLECT This person is very intelligent, but not very smart. They can best be described as the individual that knows a little bit about everything but can relate it to nothing. Their intellect, however, surfaces always at the wrong times. In an important meeting they will drop comments such as "That seems like a likely candidate for PASCAL." (Without knowing what PASCAL is or what kind of application might be suited for the language.) D.P. must then spend time negating the damage done by the statement, getting the meeting completely confused. The biggest problem with this person is that they will occasionally come up with a good idea!

THE SPECIALIST A very interesting bird! Their goal in life is to prove their way of handling data is so unique, there is no package software that will handle it. "Our payroll is just not a plain vanilla payroll. We do a lot of special things." And the wheel gets invented once again.

THE GENERALIST One of our favorite characters. Their opening comment, either on the telephone or at a meeting, is "It just doesn't work! None of it!". Upon further query, you will generally find a non-checkable data entry error, or the page numbers on a report start with page two instead of page one. In many cases, they have been looking at an old report or have just interpreted a report column incorrectly. How much time have you spent trying to determine if a problem really exists?

THE MANAGER This person has better things to do than to sit down long enough to analyze their own needs and to relate them in a design phase. They generally like to drop lines like "I need to know how many widgets have been sold in the last 3 days and the overall profit margin of those widgets in terms of the number of rainny days." Remember, you have just spent 2 years developing a comprehensive database and "rainy days" just never came up...until now. The kicker really comes when the discussion ends with "If I can't have it by tomorrow morning at 10:00 this company may lose $50,000.". When you ask about the repetitiveness of such a report, it seems that this is the most critical report the MANAGER uses, and needs it daily. I wonder what he has been using up until this time, and why it hasn't surfaced up to now?

THE EXPERT Thanks to the dropping prices in the personal computer area, everyone now has the opportunity to become a real honest-to-goodness programmer. After they have learned to write a program in BASIC to print a list of commonly used household telephone numbers for all members of the family and a program to calculate and print the batting averages for Junior's little league team, it just makes sense for them to be able to consult with data processing on how to write a fixed-asset system. How many times have you secretly wished you could get him to try to do the whole thing, just to teach him a lesson in being humble?

SOLID SAM They are most noted for their professionalism and attention to detail. When they say something will be done on Friday you know they will make every effort to have it done accurately and on time. They are gracious and make constructive comments. Because they are open to suggestions and have a lot of patience the project seems to drop into place with a minimum of effort. Fortunately there is a little of SOLID SAM in everyone.

YOUR BASIC DATA PROCESSORS

Oh, yes! Although obviously much better than the end-user, there are a few strange creatures floating around the confines of the data processing department. We can even talk about them in hushed tones...

THE BEAST This person is most dangerous when injured. Just prove that his code is really at fault...then duck. When did you say you needed that next report???

SOLID SALLY Need a complex program done in a short period of time? This is the person. Why is is that every department only has one of these? Why is it that the end-user never has one of their programs assigned to this person?

THE GENIUS Have you noticed there is always an individual living more "inside" the computer than "outside"? Their conversation is spiced more with "bits", "maps", "registers", "stacks" and other assorted strange words than we can understand. What's really scary is when you listen to two of these people when they get together for a technical discussion.

THE CODER These people are generally classified as the "entire D.P. department" by the end-user. They can even be sub-classified: The GOERS, who will graduate quickly into SOLID SALLY's; the HOPERS, who we all hope will turn into a GOER; and the WISHERS (We wish they were just a little less productive so we could fire them!).

THE MANAGER The prince of all persons. Knows nothing but can keep the balls bouncing. This person will never bet on anything without hedging. You can recognize this individual because they only use terms like "Assuming there are no problems..." and "If the user will do their job correctly..." and "Unless some unforeseen problem arises...".

THE "REAL PROGRAMMER" No one really is able to identify this person until they leave the company and someone needs to make a modification of one of their programs. In COBOL, their Procedure Division contains only one period per paragraph. Nested "If--then" complex statements abound. A simple change turns into a nightmare...and possibly a complete re-write.

THE STUDENT This is generally an individual that has just graduated and is hired as an entry-level programmer. They have just used the most exotic language and have ALREADY written a program to print address labels. They don't understand why the real world has not caught up with academia. "COBOL is such a droll language!".

THE USER SEEN THROUGH THE EYES OF THE USER

"As the person responsible for my job, I know what I do and how I do it. I know the problems involved and how to work around them. These problems are adequately documented and, through a variety of memoranda, I have informed necessary individuals that the problems do exist. In some instances, I have had some help in solving those problems, but generally I have to solve them myself.

"Given a situation, I can give data processing all of the detail they need to aid in the automation of the situation, including all of the forms involved, the data flow within my department, any exceptions that may arise, and all report and screen layouts.

"All of my documentation is up-to-date and all of my staff is trained. Procedures manuals are current."

DATA PROCESSING AS SEEN BY DATA PROCESSING

"As the person responsible for the design, coding, testing, and ultimate implementation of your project, I can assure you that I am well trained. I have had extensive background in all phases of programming, including database management.

"I am able to relate to the end-users and to understand their problems in order to solve them.

"I work efficiently and make the best use of my time, the end-user's time, and the computer's time. I am able to complete projects on time and under budget, so long as the user does not create problems or make unnecessary changes at critical stages. I am self-motivating and am able to keep up my end of the work load.

"I belong to an elite group of society because I am in the field of 'HIGH TECHNOLOGY'. Although everyone believes we all make big bucks, I am not about to tell them it ain't necessarily so."

THE USER AS SEEN THROUGH THE EYES OF DATA PROCESSING

"The user is generally involved with crisis management, continually trying to solve the current problem at hand. They are too close to the trees to see the forest. For that reason they need the services of a forest ranger...me. I can patrol the forest without being concerned with each tree, giving me a much broader concept of the whole situation.

"Because of their crisis management mode they are loosing track of the individual procedures that they set up in the first place. Their procedures manuals are outdated and are seldom referred to. They have become unorganized over a period of time and lack the discipline needed to bring it back. I can really help them better than they can help themselves.

"When they try to explain their viewpoint of the problem, they tend to be fragmented in their thought process; again the giveaway of their lack of discipline.

"They try to tell me they know what they want, but it keeps changing with each discussion. When I give they what they want they change it. I have the feeling that we could continue from now until doomsday and they would never get the report they want because of their indecisiveness. It's really a shame because I can help them to be much more productive."

DATA PROCESSING AS SEEN BY THE USER

"Those folks talk gibberish all of the time instead of plain English. They don't understand the way the company runs and they are too bullheaded to admit it. My problem is quite complex and they want to oversimplify it. If they would just listen they would see I'm right.

"It seems like I can never get anything done on a timely basis. If I have a project I have to have the approval of everyone from God on down. Then we are put on a waiting list that stretches anywhere from six months to 1998. There is a great deal of politics in determining whose project gets the highest priority.

"I have a cousin that has a personal computer and is writing programs all the time and he can get things done is a day or so. Not our programmers! The simplest of programs takes them three to four weeks..if they can get computer time. Which brings up another point: the computer is always down! Every time I need something, the damn thing is down. Its a wonder anything ever gets done.

"Those people are always responding to the latest crisis. Everyone knows crisis management is the worst kind and it has become a way of life with them.

"They keep telling me what they can do to help, but where is it? If you ask me, they are all a bunch of high priced con-artists. What I couldn't do with this company with their budget....."

THE USER'S PROBLEM

How does data processing feel about their knowledge of the user's problem?

"I CAN SEE WHAT NEEDS TO BE DONE. THE SOLUTION IS VERY SIMPLE.

How does the user feel about their knowledge of their problem?

"MY PROBLEM IS VERY COMPLEX, AND THE SOLUTION INVOLVES MANY ASPECTS, AS I HAVE EXPLAINED TO DATA PROCESSING. I UNDERSTAND THE PROBLEM VERY WELL. AFTER ALL, I LIVE WITH IT EVERY DAY.".

How does data processing feel about the user's knowledge of the problem?

"IF THE USER COULD JUST UNDERSTAND THAT A COUPLE OF MINOR PROCEDURAL CHANGES ARE NECESSARY WE COULD GET THIS SHOW ON THE ROAD. THEY JUST CAN'T SEE WHERE THEIR REAL PROBLEM IS!".

How does the user feel about data processing's knowledge of the problem?

"THEY ARE OVER-SIMPLIFYING THE PROBLEM. IF THEY WOULD JUST LISTEN AND FOLLOW MY SUGGESTIONS WE COULD GET THIS SHOW ON THE ROAD".

...AND ONE TO GROW ON!

And just to complicate the basic issue, we can see a few more little problems that crop up:

If the project at hand is a conversion or a re-write we can look at the old system vs. the new.

OLD = manual; NEW = batch This type of change is the easiest one to handle because computer files are not affected directly by the input of data. Validations can be done after-the-fact, and files can be updated later as well. This type of effort will generally take the least amount of time because the user is controlling the information through most, if not all, of the processing, not unlike their old manual procedures.

OLD = manual; NEW = real-time This is next in line for the level of easiest to design and implement. It is also the most explosive in terms of impressing the user. Unfortunately, there are not enough of these applications.

OLD = batch; NEW = real-time At this point we are confronted with all of the normal problems and, in addition, a brick wall has been constructed that will come into play during the entire project. The user is used to getting data from the computer in a particular manner and format. They have gone through the growing pains of "understanding" the way the computer people think and have gotten what they want. Now they want to apply all they have learned to this new process. Data processing now has to construct an "unlearning" process and replace it with the newest method of thinking. By far, the mental process of completing a project in this mode has extended the level of difficulty multi-fold. Unless, of course, you keep all of the old batch logic...(shades of tyranny!).

TIGHT UPPER MANAGEMENT This type of project management comes into play when top management wants to be kept up-to-date on all detail, but refuses to understand it. Much time is spent simplifying and re-writing to keep this type of management happy, generally producing reports they will not use and are of no particular value to other end-users.

LOOSE UPPER MANAGEMENT This type of leader is looking at the single bottom line: "When will the project be finished?". For the most part they will not be happy with the answer and cannot understand why the estimated date was not met. Nor do they want to learn ("I don't have the time!") anything about the project, although they will pick up on some buzz words.

AT LAST: THE ANSWER

What all of this leads up to is that all of us are people, and, for all of our strengths, will be constantly in a state of flux. We all make mistakes (Gads!), we all forget from time to time, and we are all subject to pressures.

There is a basic fact underlying any project that involves the end-user and data processing: The end-user knows his business the best and data processing knows their end the best. This sets up a natural confrontation, and the answer to the original question "IS THE FIGHT REALLY NECESSARY?" must be answered

'YES'.

But that's really too simple. Together, the end-user and data processing, will determine the severity of this battle. All of us work in strange ways to lessen the intensity because "In our hearts, we know it's right.".

Although there are no sure-fire solutions to this problem let me pass on a couple of simple suggestions that you might have tried and discarded or are still using. There are obviously no guarantees...

1. LISTEN Nothing will teach you about the other's business quite like listening with an open mind. The more you learn the easier the project really begins to get. The easier the project the less the inflamation between people becomes. Listening, however, is much more difficult when your mouth is in motion... As my favorite philosopher once said "HE WHO LISTENS, LEARNS. BUT HE WHO SPEAKS DOES NOT NECESSARILY TEACH!". (T.A. Elliott)

2. KEEP IN CONSTANT CONTACT This is a mode that can be easily implemented by data processing. It is designed to reduce the physical gap between the two parties. Take the programmer(s)/analyst(s) working on the project and require a daily contact with the end-user(s) most involved with that particular aspect of the project. This contact should be in the form of an eye contact as opposed to a telephone call. The purpose of this contact is to allow the user to be around the d.p. staff and be more comfortable with them (and vice-versa). Ideas will flow more evenly, difficulties will be headed off quicker, and the fight will be held to a minimum. The idea is to create the OPPORTUNITY for discussion.

3. KEEP THE COMPANY GOALS Above all, remember that you both work for the same company and the company is the one that is intended to profit by the project. The longer and more difficult the project is made by this battle the more the company loses. When the heat of battle begins to remember "WE ALL WORK FOR THE SAME COMPANY" and see if that realization doesn't change the temperature a little bit.

THE COST OF THE WAR

But maybe we need to ask just one more question:

"WHAT DOES IT REALLY MATTER?"

The answer to this one lies in the determination of the cost of the war.

As in the case of any war there are no real clear-cut winners. Both sides lose something, and in this case, not only are the two sides losing but the company will tend to suffer as well.

The question is "How much?".

Given that both sides have good intentions, there will be a solid amount of misunderstanding. The same language is not always used and the results of a discussion do not always lead to the desired results. Regardless of the reason for the misunderstanding, the bottom line is that famous "GOTCHA"! And, as you will find in several of Murphy's laws, the GOTCHA will never appear before the fact, but must appear immediately after that portion has been coded, tested, and approved by data processing for release to the user. The scenario that follows the GOTCHA generally requires re-coding, etc. Thus, the one single factor that seems to plague all projects has been directly affected by the GOTCHA...and that is TIME

Someone with a greater insight than this author said it first..."TIME IS MONEY".

The cost of the war is then based upon the number of "GOTCHA'S". Each one is a skirmish of one degree or another. If the number of confrontations can be reduced the cost will be cut.

So it is, the people, the battle, and the goals. If all else fails, take you enemy out for a beer....

Maybe we have been able to look at our users and ourselves in a little different light. By doing so we may just be able to deal with the other person just a little bit easier than we did before. Of course, none of this has anything to do with me... I do everything right. But for you....

INTRODUCING NOVICE USERS TO FRIENDLY, ONLINE SYSTEMS THROUGH GAMES: AN EXAMPLE USING DUNGEON, MANSION AND ADVENTURE

by
Terry and Nanci Floyd

INTRODUCING HUMANS TO MACHINES

In this age of computers we are faced with integrating their use into everyday life. This is not difficult with young students, but can be exceedingly frustrating with those more set in their ways. There are always problems when trying something new; people tend to be apprehensive of using new tools. This is especially true of computer training. In an independent survey, we heard several reasons for this apprehension.

Fear is perhaps the most frequently stated hindrance. It seems to be manifested in numerous ways. People feel that what they don't understand, they don't control. Faced with an imposing "micromonster" some people suffer from a resistance to success. The same is true of people who are afraid of trying and then failing. They are afraid of crashing the system. Other insecurities include the fear of being replaced and of not being able to learn.

Computers are seen as an all knowing entity and people feel inadequate when dealing with them. Others feel that if they don't try, they won't fail, and that guarantees safety. Some are afraid of learning something new that will change their routines. Another big concern is that computers are seen as dehumanizing. Contrary to these feelings, Marguiles(1980)[1] has argued that the computer revolution offers an opportunity to reverse certain trends of the industrial revolution that tended to result in dehumanizing jobs. He feels that being free of menial and repetitive tasks, a person would be free to be more of a supervisor and guide of the machine.

Fear is not the only thing that stops people from using computers. Documentation is sometimes a big obstacle. In our survey, documentation was often cited as being hard to understand, not having enough user examples, lack of explanation and illustrations, and just being hard to read. Most people said that having on-line instruction would be the best way of learning. Games provide the opportunity to explore this techique of teaching[2].

Not far removed from the fear of failure is the lack of trust some people have in computers. They don't trust the results they get back and are skeptical that a machine can be smarter than a human. It's a technical inferiority complex which must be overcome.

A lack of specific skills is another area of concern. Many people in the job market today have never needed to type before the office went automated. These people are now faced with having to learn a new skill and it is time consuming. They need a use for the computer that is enjoyable, but at the same time an opportunity to practice their typing. An inability to read manuals is also a stumbling block. Having clear, precise steps to follow is a must in being user friendly. Lacking motivation to care about the quality of the information being inputed is a problem. People need to be concerned enough to notice mistakes or not make them. Fortunately good on-line systems catch many errors before they become disastrous.

Computers are having an impact everywhere. When they are first introduced into the office, secretaries are supicious, especially if they threaten their jobs. Managers are fearful of what they don't understand and are scared to death of them (Stout, 1983)[1] This is not just true in business, but in schools also. Teachers in one study were found to have considerable negative feelings toward computers. The study, conducted by The Center for Automated Systems in Education, Southwest Texas State University shows that teachers need to develop a more positive attitude toward using and teaching with computers[3]. It is this attitude that we would like to see changed with the use of games. If people could sit down, logon, and have a quick response to their input, they would be encouraged to continue. Having a purpose and getting feedback quickly is a key to establishing rapport. Another good teaching device of games is it's on-line documentation. Users learn to read the screen, and learn basic commands by simply using them frequently. This interaction sets the stage for using the computers in their work. They learn to trust the computer and rely on it for quality output. The majority of people I talked with said that having indiviual training would be the best possible way to learn. This surrogate teaching aspect of computers will fill a big need in personal training.

When the next generation of computer literate children reach the work force, we will see a more open, positive attitude prevail[4]. Computers will be as necessary and as common as electricity and telephones (and as taken for granted). Until that day, we must deal with the apprehensions and fears of those leading the way.

OVERVIEW OF ROLE GAMES

Computer programs are abstract representations of reality. Introducing people to the use of computers in their everyday activities means relating their concrete experiences to symbolic representations. Somewhere between the actual physical experience of an event or situation and reading or hearing about it is the area where games exist: Mediated Experience.[5] The communications media include television,movies, still photographs, recordings, speech synthesizing chips, and graphics generators using computers of all sizes; methods of reproducing events from the past and creating events which haven't happened yet.

Games are a simulation of life. Cards and chess pieces have kings and queens. Board games have places to move and competition and challenge at every turn. Dominoes is a game of mathematical skills. Dice is a game of chance and probability. Scrabble requires language skills.

Board games like Monopoly, Clue, and Life are examples of role games. Each player is a realtor, capitalist, detective, spouse,millionaire, or pauper; one is the banker or 'master' of the game. When the concepts behind such games were combined with the power of computers to 'simulate logic', produce sounds, graphic outputs, and print words on screens, role playing computer games were born.

On the HPIUG Contributed Tape Library are several intricate role games: Adventure[6], Mansion, and Dungeon. Unlike arcade games, such as PACMAN, DONKEY KONG, and ZAXXON[7], these computer role games require more thought than physical reflex movement.

The head of IBM's Personal Computer operations says, "Games aid in the discovery process[8]." Children obviously learn about life in the games they play. Atari, Mattel, and Coleco have brought computers into the homes of millions more people in the last few years than business, governmental, financial, and educational organizations have in the prior thirty years. There is a message here, but computer costs have decreased to a very low level during the same period.

In studies of "human factors associated with the use of computers for instructional use, games packages have offered a new avenue for research into areas of problem solving using complex visual displays[9]." Such studies should also apply to the use of computers for non-instructional purposes and the display of natural language (wriiten or recorded) media.

Role playing games foster an involvement in a fantasy. There is a common theme of survival against great odds amomg Adventure, Mansion, and Dungeon. These games are really computer programs that describe geographic locations (forests, rooms in

houses, caves, and passages) and simulate in words reactions to the player's moves. Besides endless descriptions of places, these games' protagonist (is it the computer? or is it the programmer who wrote the game?) leads the player through deadly situations involving warriors, dwarves, knights, kings, pirates, wizards, giants, trolls, and beasts of all sorts weilding knives, hatchets, guns, swords, clubs, and other dangerous weapons.

Imagination is the primary prerequisite in a role game player (adventurer). This fact may shed some light on the existence of a high percentage of science fiction freaks who have overcome their fear of computers to the point of addiction. People who refuse to participate in the game fantasy are likely to have trouble relating the use of computers to their jobs.

Since these role playing games have no user documentation, they reflect reality where the perfect documentation has yet to emerge. The game programs themselves tell you just enough information to solve each problem. The player has to make discrete inquiries through typing commands to the computer to discover what will cause the desired result. Although 'HELP' is a valid command, it usually produces sarcastically obvious information from the all seeing 'gamesmaster' the computer program represents.

The novice player quickly gets used to thinking and reading the dialog carefully before typing in a command or response. Players frequently talk to their terminal for hours, too often cursing, while getting 'killed' several times at the same point in the game.

The games are similar in that the player wanders around the landscape collecting and defending prizes (coins, jewels, matter transfer devices,food and drinks, and weapons). Adventure, the earliest of the role games, has an important feature built in - a clock and security system to prevent long games during working business hours. This is of course important when dealing with the games enthusiasts mentioned earlier and the degradation of system response time when too many players are trying to share too few resources with other users doing invoices, purchase orders, and payrolls.

Dungeon is loosely based on D&D, the Dungeons and Dragons game (see Rick Mc Leod's paper, attached[10]). A primary purpose of Dungeon is to collect prizes during trips through a dangerous cave and to put them into a trophy case in a small white house. The entrance to the cave is through a trap door under a large rug in the living room of the house, but there are also lots of frustrating piles of leaves outside with locked grates leading downward. Dungeon, more than Adventure or Mansion, requires the player to engage in battle with many hostile oponents while the program monitors the player's strength, deciding if he has enough power to defeat the attacker.

The purpose of Adventure is to visit the many rooms and crawlways of Colossal Cave, collect all the prizes, wait for the 'end game,' and survive Witts End. This can be accomplished by an almost perfect adventurer in 30 minutes or less. Others have spent weeks at the 'rank amateur' level of competency.

Mansion is the computer role game version of the board game Clue by Parker Btothers. The player ventures through a MYSTERY MANSION trying to get the murderer and the weapon to the scene of the crime where the corpse is lying in a pool of blood.

The first obstacle is the gate in front of which you have been left by a taxi. Chasing the taxi is a waste of time, and time is critical. It is dawn (6AM); the sun sets at 6PM and the mansion explodes (subversives in the basement?) and burns to the ground at midnight.

You can see an old mansion just visible through the gate. A road runs east and west along the wall into which the gate is set. The program explains all of this and more to you, then without further instructions, awaits your move.

"WHAT DO YOU EXPECT TO HAPPEN?" you type, punctuating with the RETURN key.
"I WILL NOT ANSWER ANY QUESTIONS. I ONLY RESPOND TO YOUR ACTIONS." This is a character mode dialog.
"HELP," you type on the keyboard.
"HINTS COST 20 POINTS. DO YOU WANT ONE?" comes the prompt.
You resond, "YES."
"A GOOD WAY TO START IS BY SAYING (sic) 'GO WEST' OR 'GO EAST.'"
You, being clever speak aloud, "GO NORTH." The cursor blinks; nothing happens. Then you type it.
"THE GATE IS CLOSED," says the computer.
"OPEN THE GATE," you type as you think how easy this is going to be.
"THE GATE IS TOO HEAVY FOR YOU TO OPEN ALONE."

Thus you face your first dilimma. What to do?
"CLIMB GATE," is your bold command.
"THE GATE IS COVERED WITH ALL KINDS OF POINTS AND BARBS AND YOU WOULD NEVER MAKE IT OVER ALIVE."

"GO SOUTH;" no use trying to get in there. Maybe the taxi's at a bar down the highway.
"YOU ARE ON THE HIDEOUS HIGHWAY." Can you see the heat waves rising from the pavement?
"THE HIGHWAY GOES STRAIGHT SOUTH AS FAR AS YOU CAN SEE BETWEEN THE DENSE FOREST TO THE WEST AND A 500 FOOT SHEER CLIFF TO THE EAST. THE MANSION GATE IS ABOUT 200 YARDS TO THE NORTH."

With cliffs, forests, and an endless highway, you must go back to the gate and check out the road that goes east and west by the wall. Back at the gate, there's a note on the ground; pick it up, it's documentation! Read it!

After solving the wall problem and several others, you will eventually find yourself inside the mansion:

"YOU ARE IN THE ENTRANCE HALL. THERE IS AN OLD CRANK TELEPHONE ON THE WALL BY THE FRONT DOOR. A LARGE WINDOW OVERLOOKS THE GROUNDS IN FRONT OF THE MANSION. THERE ARE FOUR DOORS SO THAT YOU CAN GO LEFT, RIGHT, FORWARD, OR BACKWARD."

"GO NORTH," it worked before, didn't it?

"YOU NEED A COMPASS TO TELL CARDINAL POINTS HERE," comes the curt reply.

"GO RIGHT," you concede.

"IT IS DARK HERE. A CURTAIN BLOCKS THE LIGHT." Now you wish you'd gotten that lantern back at the front gate.

"OPEN THE CURTAIN," you demand.

"YOU ARE IN THE GAME ROOM." (How appropriate).

"YOU CAN SEE: A SWORD AND A DUSTY GLOBE. THE ROOM IS DILAPIDATED AND HAS THE REMAINS OF VARIOUS OLD STYLE (and unused) GAMES SUCH AS DARTS, DUCKPINS, SEVERAL CARD GAMES AND AN OLD RADIO. THERE IS A STAIRWAY GOING UP AND DOWN, AS WELL AS A DOOR YOU CAN EXIT BY GOING BACKWARD."

That is a lot of information to digest. Since these descriptions are the only documentation you have to go by, what do you try first?

"GET THE RADIO," you guess.

"YOU CANNOT CARRY THE RADIO." Rats! Well, at least you didn't wreck anything. You also learned that the computer understands the command 'GET.'

"GET THE SWORD AND THE DUSTY GLOBE," you say wondering just how smart this thing is.

"MY INPUT VOCABULARY DOESN'T INCLUDE THE WORD DUSTY."

Pretty smart program, huh? "GET GLOBE AND SWORD."

It works: "YOUR BOOTY NOW CONTAINS THE GLOBE AND THE SWORD." My dictionary defines booty as "plunder taken from an enemy in times of war; seized or stolen goods; or any valuable prize." You begin to wonder about the character of the role you've been dealt. Are you a soldier or a pirate? If so, you must be winning the game.

"SCORE," you enter, hoping it's a valid command.

"SO FAR YOU HAVE SCORED 70 POINTS. YOU CAN STILL REACH 999 POINTS." Maybe you're not doing as well as you thought.

"BACKWARD," is all you can think of to do as paranoia takes control.

"YOU ARE IN THE ENTRANCE HALL." Well, that makes sense.

"GO RIGHT."

"IT IS DARK HERE." Uh-oh.

"GO FORWARD."

"THE TOWER BELL RANG 7 TIMES." It is 7AM.
"IT IS DARK HERE, A CURTAIN BLOCKS THE LIGHT."
Feeling claustrophobic, you "GO LEFT."
"YOU ARE IN THE CHARMING CHAPEL." Ah, a sanctuary, you hope as you read on... "THERE IS AN ALTAR IN FRONT OF THE CHAPEL WITH SOME COMMUNION WINE ON IT. THERE IS ALSO A BURNING CANDLE ON THE ALTAR ABOUT 5 INCHES LONG. A SILVER CROSS ABOUT A FOOT LONG HANGS ON THE WALL BEHIND THE ALTAR. THERE IS A STAIRWAY LEADING DOWN INTO THE DARKNESS AND A WAY TO GO BACKWARD THROUGH AN ARCH WHERE YOU CAN SEE PART OF A LIBRARY."

You've certainly discovered a lot of stuff here! You decide to start with the candle.
"GET CANDLE."
"YOU BOOTY NOW CONTAINS THE CANDLE." The program never tires of telling the player the details of mundane events. It is important to notice however, that a long description of a place is printed only the first time the player arrives there. A short description prints the next and all subsequent times. To obtain the long description again, the player must type "LOOK." This means the program is keeping track of what it has told the player and expects him/her to learn.
"GO DOWN" into the darkness, you command, balancing on that line between bravery and foolishness.
"YOU HEARD A WOMAN SCREAM." Oh-no. Someone's been hurt.
"YOU ARE IN THE STORAGE ROOM. THE ROOM IS CLUTTERED WITH MOSTLY USELESS JUNK. THERE IS A LARGE INTERESTING LOOKING CRATE HERE. YOU CAN GO UP A WELL WORN LADDER OR BACKWARD THROUGH A HEAVY DOOR WITH A SMALL WINDOW WITH IRON BARS. THERE ARE ALSO SOME STEPS CUT IN THE ROCK LEADING DOWN."
You try, "LOOK AT THE CRATE."
"I DON'T KNOW WHAT YOU EXPECT TO SEE." The program has assumed a first person approach to the player.
Well, that didn't work, how about "MOVE CRATE."
"I DON'T KNOW WHAT YOU EXPECT TO HAPPEN," says the unfeeling program. How depressed it sounds to the player who has left the familiarity of his surroundings and is now really engrossed in the story unfolding within his and the computer's minds.

Frustrated, the player enters, "OPEN THE CRATE."
Success: "YOUR ACTION OPENED A PANEL INTO THE SECRET PASSAGE THAT YOU CAN ENTER BY GOING LEFT."
Of course, you "GO LEFT" to find "THE CORRIDOR WALLS HAVE OPENED, YOU ARE IN THE SECRET PASSAGE. THERE IS AN AWKWARD STAIRWAY LEADING UP TO AT LEAST TWO OTHER LEVELS. YOU CAN GO IN ANY DIRECTION BUT DOWN HERE."
"GO LEFT." Why not?
"YOU ARE IN THE CREEPY CRYPT, YOU CAN SEE AN OLD TALISMAN. YOU ARE HERE WITH THE BUTLER AND THE VAMPIRE." This doesn't look good at all. "THERE ARE MANY COFFINS HERE CONTAINING THE REMAINS OF THE ANCESTORS OF MYSTERY MANSION. ALL THE COFFINS APPEAR TO BE UNDISTURBED AND SEALED EXCEPT FOR ONE. UNLESS YOU CAN OPEN A SECRET PANEL, YOU HAVE TO GO RIGHT TO GET OUT OF HERE."

The situation looks pretty grim what with the vampire and all these coffins, so you pray and type in "GET TALISMAN." It's your only hope; maybe it's magic.

"THE VAMPIRE HAS ATTACKED YOU AND SUCKED THE BLOOD OUT OF YOU. YOU DON'T HAVE ENOUGH POINTS TO REINCARNATE. YOU SCORED 84 POINTS WHICH RATES YOU AS A PITIFUL SLEUTH. YOU PLAYED 25 MINUTES REAL TIME AND 2.6 HOURS GAME TIME OR 75% UTILIZATION. BETTER LUCK NEXT TIME. BYE BYE."

END OF PROGRAM.

The average player will want to try again immediately. Most people spend several hours before they give up or find someone who knows enough of the game layout to help them survive long enough to get 'really into it.' Just as there are some users who don't see any point to these games, there are those who spend every waking minute struggling with the simulated situations which get ever more complicated.

CONCLUSIONS

While not all novice users are afraid of computers, those who are can erode the effectiveness of any on-line system. Maybe the point has been made that computer games can ease the transition of adults from novice to competent users of computer systems. The study of how these games motivate and educate their players may also be of use to system designers and programmers. The future holds much promise for the integration of documentation and procedures within programs.

Imagination is the critical resource on the data processing side, when designing systems, just as it is to the adventurer when trying to decide how to solve a role encounter. The user's role is to play the game, the programmer's role to make it as easy as possible to learn and use. If some of the 'fun' of computer role games can be transferred to 'friendly on-line systems,' some of the boredom inherent in 'feeding data to the computer' might be releived. Removing some of the repetitiveness from data entry could ease the fears of those who say computers are dehumanizing.

The disadvantages of computer games are addiction and cost. Some people neglect reality in pursuit of the never attainable victory over the computer. Apparently very few adventurers reach the ultimate conclusion of the game. Maybe that's not so bad when one considers that people have been known to play Bridge (a card game) for a lifetime, discussing hands played with friends in years long past.

Computer games cost money as well as time. At 2PM on Wednesday, your average HP3000 has very few disc I/O's to spare for a game of ADVENTURE. If the computer happens to be in California and the terminal is in Texas, someone or some company is

paying a pretty large phone bill to be able to "KILL THE WARRIOR." Is it worth the expense?

The answer to this question can only be determined after considering any advantages of computer games. The need for human beings to feel comfortable with machines so as to use them more productively is one conclusion. The lessons to be learned by software and hardware designers is probably of more significance. If the programs people use in their everyday tasks can be made more friendly and amusing, their fears can become a willingness to participate.

With the attention of both software and hardware designers turned toward producing more user friendly machines, one hopes that humans will become less machine hostile. It might be a good start to stop instructing (even in written documentation) the novice to "HIT" the carriage return key or "DEPRESS" the shift key. The poor machines are already depressed without being hit. But all that may change (as soon as the computers begin talking back).

Ergonomincs is the label that has been given to the study of making work comfortable. A great deal of effort needs to be put into programs to make instructing computers a more humane task (as hardware is being designed to humanize procedures). If keyboards can be standardized to the point of having numerous named function keys (like 'BACK-UP','PRINT','HELP','MENU', 'UNDO', and 'DRAW')[11] then why can't parsers (those pieces of code in programs that figure out what you're trying to do) evolve a standard set of commands? Why not use English?

In 1973, James Martin said natural language is too complicated to be used for the human-computer dialog [12]. Yet, role games seem to be using natural language. If commercial users could be challenged and motivated like game players are, the task of software usability would be easier. But, getting 'killed' in Mansion because the player tried the wrong verb is not like crashing the payroll run because of an input error. Or is it? Users are, after all, responsible for their actions just as game players are. As John M. Knapp[13] points out in his review of ergonomics, people speak different dialects of a language(including jargon)[14]. The better the programmer understands his/her audience's language (dialect), the easier their programs are to use.

Glenney and Rylander summed it up well..."The proliferation and productivity of the industrial work force...[will be complete] when the fundamental issues of how management employs the effective use of capital in implementing computer integrated systems to stimulate the commitment to providing an environment in which employees can direct themselves."[15] They go on to say that providing these tools can "insinuate additional incentives for a person's work performance....It is this personal motivator in the workplace performance that generates achievement."

BIBLIOGRAPHY

1. Time Magazine, January 3,1983, p. 14-24.
2. Nickerson, Raymond S. "Why Interactive Computer Systems Are Sometimes Not Used By People Who Might Benefit From Them," INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES (1981).
3. Stimmel, Connor, McCaskill, and Durrett. "Teacher Resistance to Computer-Assisted Instruction," from BEHAVIOR RESEARCH METHODS & INSTRUMENTATION, 1981, Vol 13(2).
4. Computerworld, February 7, 1983.
5. Dale, Edgar. AUDIO VISUAL METHODS IN TEACHING, New York. Holt,Rinehart, and Winston, 1969.
6. Crowther, Willie and Woods, Don and Palter, Gary. Developers of the ADVENTURE GAME.
7. Copyright Midway, Nintendo, Sega.
8. Time Magazine, January 3, 1983, p. 16.
9. Durrett, John H. and Zweiner, Catherine. "A Microcomputer-Based Laboratory: Real World Experience," ibid(3).
10. Mc Leod, Rick. "Using Games for a Friendly Introduction to Computers", attached.
11. Rutkowski, Chris. "An Introductoion to the Human Applications Standard Computer Interface, Part 2: Implementing the HASCI Concept," BYTE, November 1982.
12. Martin, James. DESIGN OF MAN-COMPUTER DIALOGUES, Prenitce Hall, 1973.
13. Knapp, John M. "The Ergonomic Millennium," INTERACT MAGAZINE January/February,1983, p. 57.
14. Heckel, Paul. "Maxim for 'Intelligence Design': Know Your Users," INFOWORLD, July 19, 1982, pp. 11-12.
15. Glenney, Neil and Rylander, Robert. "Real Time Shop Floor Control," PRODUCTION AND INVENTORY MANAGEMENT, Fourth Quarter, 1982, pp. 121-139.

Using Games for a Friendly Introduction to Computers

Rick Mc Leod

Rick Mc Leod
Hewlett-Packard
19447 Pruneridge Ave.
Cupertino, CA 95014

I. In recent years, newpaper articles which chronicle a new type of computer crime have appeared. These crimes are not committed with the computer, but to the computer. Frustration and fear have led otherwise normal people beyond banging on the keyboard, now, they pour Pepsi through the fan grill, push their terminals off the table, etc. These extremes are not often reached, but it is distressing when they are.

Why should these things occur? One never hears of a craftsman throwing his powersaw into the ocean. This is because the craftsman understands and respects his tools, and he enjoys working with them. If this same attitude can be fostered in the non-technical user, then a happier and more productive employee will result.

The major fear facing the wouldbe user results from identifying the computer as a sentient being designed to make him miserable. This perception develops in many ways, one being that it is hard for a non-technical person to believe that his credit rating and insurance have been fouled by 'an overgrown adding machine.'

The first step towards altering the user's attitude is changing his perception of the machine. The user must be properly introduced to his new tool, and be made to understand that the tool isn't really human, even though we may design them that way. The game presents an ideal format for this perception change.

The game takes the user's mind off the phobia and puts him in environment where he feels comfortable. Witness the popularity of the video arcade game and the home video game. These have allowed micro-computers to move smoothly into millions of homes with very little apprehension on the part of the buyer. In 30 years, very few people will have technophobia because most of them will have been taught by computers at some point in their lives.

Once the user's concentration has shifted from the phobia, the battle essentially is finished. The computer has changed from a hideous monster to a child's plaything and a smart one at that. The user can then better understand the computer and how to use it efficiently and effectively.

There exists a second fear that belongs not to the user, but to his manager, the fear of the game. If the idea of placing games on the computer is even hinted, a phantasmagoria of images swirls like a tornado through his brain, visions of people slacking off work, entering caverns instead of purchase orders and the like. This conception is not entirely unfounded. If games were unsupervised, it is doubtless that some people would not work at all, however, with proper management, games can coexist on the system.

II. There are two major classes of single player games, the strategy game, in which the computer is the opponent, and the role-playing game, in which the computer acts as the players guide.

Most people are familiar with the strategy game. Chess, qubic, reversi (e.g. Othello) are common examples of strategy games. These games are most useful in that most are already familiar with them, and thus do not require any special knowledge other than how to logon and enter data. They are also effective in acquainting the user with the terminal keyboard and certain types of interfaces that may be encountered in a program.

As a general rule, strategy games can be played in a short amount of time, or they can be ended without the loss of too much 'play effort' if it should be necessary to abort the program. For this reason, they make excellent lunch fillers.

The second class of game, the role-playing game, developed in the late 70's as an offshoot of the game, Dungeons and Dragons. D&D, as it is referred to among its players, was the first in a new concept in games. After its publication in 1973, several computer science students began attempting to simulate the game to some degree. The most popularly known success was Adventure by Scott Adams, (see paper by Terry and Nanci Floyd).

As parsing theory became more advanced, games became 'smarter.' The concept of Adventure evolved into Dungeon. Adventure was limited to commands of one or two words of six letters or less which took the syntax form of <verb> or <verb><object>. Dungeon introduced adjectives and prepositions to the command language and greatly expanded the possibilities of the game. Proof of this can be found just by comparing commands examples.

```
Adventure:  KILL  DRAGON
           <verb> <obj>

Dungeon:    KILL  TROLL  WITH  SWORD,knife,toothpaste,etc.
           <verb> <obj> <prep> <obj>
```

The sophistication of the command structure begins to rival that of transaction processors and report writers on the market today. These games teach the user to think in a concise manner that will get the job done.

Dungeon taps and expands the creativity of the individual because the user must imagine the object(s) need for particular tasks and the commands required to perform them. Command lists are not supplied with RPG's. The player must make logical connections between places and objects, objects and messages,etc., solve increasingly complex puzzles, and 'stay alive.' The feelings of accomplishment and satisfaction come upon passing to higher levels of expertise which is also desirable.

III. There is still the problem of providing automated supervision to the games which, indeed presents no problem at all. This describes several methods for limiting access to games and preventing interference with daily data processing.

First, a separate account for games should be created with two userids, perhaps MGR and PLAY. They should reside in different home groups, one for source code and the other for object code. If you are concerned with system performance, create the id PLAY with MAXPRI=DS or ES. This will keep game players out of the competition for the CPU; they will operate during otherwise idle time. The userid PLAY will require IA and PH capability. The home group for play will also require IA and PH. If Dungeon is to be installed on the system, the group and account will require PM, however, the id PLAY will not.

WARNING: Installing Dungeon on a computer without access precautions of the following type compromises the security of the entire system. Dungeon uses priv mode to access the system clock so that it can only be played during certain hours.

The groups should be created with restricted file access, and all files should be 'created' by MGR. Source files should be restricted to creator access only. Object code files should be restricted to group user:execute only status.

A UDC file must be created for the id PLAY. This UDC will consist of a logon command and a whole lot of logoff commands.

Example:

```
HELLO
OPTION LOGON NOBREAK NOHELP
TELLOP Game Session beginning (optional)
RUN CONTROL.PUB.GAMES
BYE
**
SETCATALOG
TELLOP Successful Break out of game...logging off
BYE
**
(MPE COMMAND)
TELLOP breakout
BYE
 .
 .
Ad infinitum
```

Therefore, if someone should manage to get out of a game and into MPE through a stack overflow or other program error, the operator will know, and the player will be logged off so that no damage, however unlikely, can be done.

Following is an example of the program CONTROL.PUB.GAMES. The program should be prepared with IA and PH (and PM if Dungeon is to be installed) capability. This is the bare bones of what can be a sophisticated program. Further additions which might by useful for your site are:

1. Call to DATELINE with addition control logic that will limit the time of entry to non-working hours.

2. Call to COMMAND to produce a :LISTF,0 of the object files available.

3. Overall elaboration of messages to the user, such as 'Which game do you wish to play?','I am not familiar with that game, please try another.', etc.

The program itself is very simple. It is designed to insulate the novice user from any hostility of the operating system, and to protect the system from abuse by the experienced user. The program requests a program name until either the activate call is successful or no data is entered. In the latter case, the program terminates, and in the former, it executes the requested program file and logs the session off after completion. The terminate call at the end is redundant and is provided for clarification only.

```
Program CONTROL

begin
 intrinsic read,print,create,activate,terminate;
 logical array progname(0:4):="          ";
        array errmsg(0:4):="no go, joe";
 logical fl:=%41,createflag:=false,activateflag:=false;
  integer pin,susp:=3,l;
  do begin
    do begin
    l:=read(progname,4);
      if l=0 then terminate;
      if <> then terminate;
    create(progname,,pin,,fl);
      if = then createflag:=true
      else print(errmsg,5,0);
   end
    until createflag=true;
  activate(pin,susp);
    if = then activateflag:=true
    else print(errmsg,5,0);
    createflag:=false;
   end
  until activateflag=true;
terminate;
end.
```

Other possibilities for controlling games include:

1. A job running a background process which activates an account password for GAMES when X or more sessions are active.

2. A job to further restrict file access when X or more sessions are active.

3. A job to limit the number of games sessions running or to limit the devices that can run games.

Consolidating several of these methods into a single control process provides greater control, control which is tailorable to the needs of individual installations.

IV. In the final analysis, we can see that all fear is unfounded. The user certainly has no reason to fear, get angry at/with/about, or beat on, the computer. Likewise, the manager has no reason to fear a loss of productivity/performance/output from his computer or his employees. Using games to overcome 'technophobic' anxiety can be accomplished to the advantage of labor and management.

References:

Dungeons & Dragons, by Gary Gygax. 1973, Tactical Studies Rules.

Psychology Today, various issues, May to December 1982.

SPL Reference Manual, Hewlett-Packard, 12/77.

Dungeon, Frobozz Magic Adventure Company, MIT.

Hiring Programmers for the HP-3000

Richard A. Fontaine

Vice President

IBMI

My company is a twenty-five year old Service Bureau in Washington, D.C. In recent years, we have concentrated on the Insurance industry and expanded our product line to include software sales and programming contracts. The Company's need for programmers fluctuates, but new lines average three per year.

Until three years ago I operated as an independent consultant. IBMI was one of my clients. Another client of mine purchased a HP-3000 in 1975. Like so many other people have done, I immediately "fell in love" and have worked almost exclusively on the HP-3000 ever since.

When I joined IBMI as an employee in 1980, my first order of business was purchasing a HP-3000 Series III to replace an IBM System 360 Model 50 which used a Model 30 for input/output. Computer room costs were reduced by two thirds and the new environment was receptive to the On-Line system that everybody wanted. A lot more people "fell in love".

It took approximately six months for the programmers to become comfortable and fully productive on the HP-3000. We compared this to the one-year period that is needed for programmers to understand our bread and butter application, Group Insurance Billing and Reporting. Parenthetically, group insurance is sold and serviced through the mail by independent administrators frequently in cooperation with an Association whose members are motivated to remain active members in order to retain their insurance. It is a discount product in comparison to insurance products purchased individually from a broker or an agent of an Insurance Company.

When the need for new programmers arose, we looked at the four possible choices insofar as applicable experience:

1. Experience with HP-3000 and Insurance
2. Experience with HP-3000 only
3. Experience with Insurance only
4. No experience.

It was obvious that very few job applicants would fit into the first category. Also, the first three categories all included applicants with very high salary expectations. Programmers with experience on the HP-3000 or Insurance but not both would be highly paid and still require an extensive training period. The fourth gourp offered several advantages:

1. Low salary expectations, at least initially
2. Large number of applicants
3. Training in both areas could overlap.

We decided to experiment with inexperienced programmers and devised a simple test to reduce the number of applicants to a manageable number. The test consisted of a COBOL program which I will describe later.

The first classified advertisement was a one-column, one-inch ad which ran one day in The Washington Post:

PROGRAMMER
TRAINEE

No experience needed. Must have completed COBOL training. Selected applicants will be tested. Salary $200/week with excellent opportunities for advancement. Send letter or resume to

IBMI PERSONNEL
2133 Wisconsin Avenue, N.W.
Washington, D.C. 20007

The first ad in January 1980 resulted in over one hundred responses during a time when the economy was fairly strong. I selected thirty-five people for testing, interviewed the top four and hired the top two. Both of these became very successful programmers and one of them is becoming an excellent System Analyst and project manager.

Later versions of the want ad specified "US Citizen", "Degree Required" and "Two Semesters of COBOL". As the economy worsened, the number of applicants increased despite the addition of the degree requirement and the unchanging salary of two hundred dollars a week! I should add that we have recently increased the starting salary, not to attract more applicants, but to keep the selected

applicants longer. Our policy of reviewing salaries every six months for the first two years has produced an average retention of less than two years; we expect to improve on that by increasing the starting salary so that the people are more contented in the beginning.

The test is a single COBOL program which we have found can be written by a trained but inexperienced programmer in three hours. The program has a sequential input file and prints an error list, as well as a summary table. We insist on the use of subscripting for the summary table and have found that this requirement eliminates seventy-five percent of those taking the test, and ninety-nine percent of those with less than two semesters of COBOL training.

The COBOL program is essentially PASS/FAIL graded; a perfect or nearly perfect program is needed to PASS. This makes grading the test very simple. It also makes grading the test a bit discouraging because only two to five percent of those taking the test come even close to a perfect program. It is important to be strong and to remember that there are people out there who can pass this test. The advantage of testing is that my secretary arranges the appointments and administers the test. While I do the grading myself, I have found that this requires very little of my time if I insist on perfection. On one occasion I felt rushed to hire someone and made two mistakes which should be avoided:

1. I arranged the appointments and administered the tests myself, which put pressure on me to hire someone quickly. This also gave me too much contact with the applicants. The latter led me to pre-judge the applicants and encouraged me to make the second mistake.

2. I did not reject less than perfect programs and spent an inordinate amount of time ranking imperfect programs.

There were two recruiting failures that occurred using this test for hiring trained but inexperienced programmers. One was the result of making the mistakes described above, which resulted in my hiring a very likable young man who did not have the "super-coder" aptitude that my Company had come to take for granted. When we lost a major contract and decided on a Reduction in Force, he was the first to be released.

The other recruiting failure occurred early in 1981, the second time I used the test. One of the early applicants had four months experience as a COBOL programmer and completed the test in two hours. She declined the job offer, however, because of low pay and a poor commuting situation. So I continued to test three dozen more applicants until an applicant wrote a perfect program in an hour and a half. I was elated and felt priveleged to be able to hire her. But I was in for a very unpleasant surprise. Her work raised serious questions concerning her level of aptitude and after a few months, I decided to work very closely with her on a project in order to evaluate her. It gradually occurred to me that she must have seen the test before taking it, particularly when I observed her friendship with one of the older employees, who later left the Company. She was released three weeks before her six month provisional employment period ended.

I now realize that the best potential programmer will use the full three hours, and, if finished early, will desk check the coding and the block diagram. The person who finishes early is either experienced, has seen the test before, or is less than a perfectionist about his/her work.

After the above experience, where I am convinced that the applicant had access to the test beforehand, I decided to design a new test. This time the emphasis was on single character moves using subscripts. Unfortunately, the second test was much too difficult and no one could finish it in three hours. Furthermore, the specifications were not specific enough. I have learned that writing specifications for a group of inexperienced programmers is an art. A vital part of the test administration procedure is that no questions may be asked. To be fair, the explanation of the requirements must therefore be extremely clear. It must not be ambiguous or the program will be difficult to grade. If many of the tests include notes about assumptions that were made, the statement of the problem needs more work. Many applicants will deliberately misinterpret the problem in order to avoid using instructions with which they are unfamiliar, such as subscripting and using edit marks.

After using the second test for six months, I returned to the original test and continued to revise and polish the statement of the problem. I correctly assumed that anyone who had legitimately taken the test before would either stay away or make the fact known before taking the test. Cheaters would finish too early and thereby raise my suspicions wo that the second test I devised could be useful after all.

Applicants who receive a reject letter frequently call me and occasionally visit me to review their program. I treat these people with great respect since they have given me at least three hours of their time already. I admire them for their courage in calling me, as well as their sense of purpose in wanting to know "why not?" One applicant who politely asked to see what he did wrong was hired by us as a Control Clerk and still intends to prove me wrong, in a friendly way, for not hiring him as a Programmer. From time to time, the recipient of a reject letter makes an angry call which emphasizes to me that this is a serious matter impacting the lives and careers of these people. I have found that showing an applicant his or her own test along with the program submitted by a person who was hired is the most convincing way to get the message across to an insistent applicant.

In grading the test, I look for reasons to reject the program since most of the applicants will fail and one of my objectives is to minimize the time I spend grading programs. Over half of the programs can be rejected in less than five minutes. I have found that the errors occur in the Procedure Division, particularly in the logic. Many programs appear to be very professional on superficial inspection, but contain no end of job routine. Since the requirement includes four error messages, I look for efficiency in coding the data validation which most applicants are able to program, but only the best ones are able to program with efficiency, avoiding repetitious sequences of instructions. Those who pass the test always use either a common routine, or a common closed sub-routine to list the error record with the appropriate error message on the printed report. While repetitivecoding may work, it is usually a clue that other errors will be found because the aptitude of the applicant is not as high as I would like.

The test instructions stipulate that "structured programming techniques are optional" and that applicants "will not be graded higher or lower for using structured programming". This may be heresy in some circles, but the point is that I am looking for COBOL knowledge and programming aptitude. The use of PERFORMs instead of GO TOs only reflects the instructor who taught the applicant at this point in his or her career.

The test instructions also stipulate that breaks may be taken at will, but no additional time will be given. This reflects the real work situation. I have found that most applicants will sit as if glued to their chairs for three hours. The highly restless person frequently turns in a one-page program and leaves early.

The length of the program is another indication of things to come. All of the perfect programs were written on five pages of COBOL coding paper, but the average program is eight pages long. The economical programmer seems to be the high aptitude programmer. The short solution takes less time to write and thus allows more time for desk checking.

At first I tested people one at a time and scheduled two people each day. But this is frustrating because I normally test thirty people to find one perfect solution. At that rate, it was taking three or four weeks and the temptation to pick a less than perfect program was enormous. I found a large room that could hold a dozen tables and chairs and started scheduling thirty people per day, fifteen each at nine AM and one PM. That seems to be the answer.

Like the airlines, I overbook because I know that twenty percent will "no show", but I also have a way to deal with a full house if they all come.

Testing in groups makes the grading of papers more efficient. I have found that making several passes eliminating programs is faster and enables me to get my own mind increasingly focused on the problem program. I also use a form letter which I hand address as I reject programs.

Having gone through such a process to find one or two perfect programs, I send the chosen few a letter of congratulations and ask them to call me for an appointment. A small number will have found other jobs, but most of them will call. I learned early to interview only those I had already decided to hire, and to use the interview to reassure the applicant and explain the Company. I use the interview to sell myself and encourage the applicant to do the same. This reduces the probability of being turned down. Also, some of these people have been turned down dozens of times and have sometimes been out of work for months. They need time to assimilate the idea that someone wants to hire them I have found that the longer a person has been out of work, the more time he or she will need to make a decision to accept a job offer. I have also found that some of my chosen few interview better than others. I have never allowed a poor interview change my mind and intend to continue with that one. Let me tell you why.

The programmer applicant who interviewed well will probably turn out to have more potential as a Systems Analyst or Manager. But the odds are very high that he or she will develop those skills on somebody else's payroll. When interviewing entry programmers,

I look for people who will be good entry level and journeyman programmers. The test has already made that selection, so I will continue to use the interview to sell myself and the Company.

Before concluding this paper, I want to address the matter of what to do with new inexperienced programmers. First, I have found that little is gained by giving them time to read the manuals or attend training courses. Similarly, time spent studying system description narratives and user manuals is unproductive. These people want to program, so I try to get them started within one or two days on a very simple program, ideally one that they can finish in a gew days. Other programmers help the new person to learn EDITOR, how to compile their program, and how to interface with the computer room. The next step might be a substantial program, but it is always a batch program so that IMAGE can be learned before VIEW. I encourage the new programmer to ask for help if he or she is stuck applying the rule that a good night's sleep cures many program bugs, but the second sleepless night should be shared with someone else. Sometime during the second day of struggling with the same problem, the programmer should seek help.

I have found that new programmers, hired after being selected by the test, and allowed to learn the HP-3000 gradually with little or no pressure for the first two or three months, become very productive quickly. I know that the famous user friendliness of the HP-3000 makes this possible, and I am grateful. My programmers are grateful, too.

So many people invest time and money to study Data Processing at a University, two-year college or six-month technical school only to discover the Catch-22 that is buried in so many want ads. As one young man put it, "Everybody wants you to have experience, but no one is willing to give it to you unless you already have it."

It's sad, really. It gives me a lot of satisfaction to be able to give a few people the opportunity to escape the Catch-22. With the HP-3000, why not?

PROGRAMMER TEST

1. You have three hours to complete this test program. If you are not able to finish, turn in what you have done and it will be evaluated.

2. You may consult any notes or reference material that you have with you but you should not be concerned if you have no reference material with you. You should not need it.

3. You may break at any time for as long as you wish but the three-hour time limit will not be extended.

4. You must turn in the examination three hours after starting. You may not return at a later date to complete it. If you can not stay here three hours today, stop now and make an appointment to take the test some other time.

5. The examination consists of one COBOL program. You should read and understand the program requirements, prepare a free hand logic diagram, flow chart, or block diagram, and code the program on COBOL coding forms.

6. Structured programming techniques are optional. You will not be rated higher or lower for using structured programming.

8. The attached sample program is not part of the problem. It is included to save you time in writing the first few statements such as IDENTIFICATION DIVISION, etc.

TUE, JUL 27, 1982, 1:29 PM

PROGRAM REQUIREMENTS (August 1982)

1. GENERAL. Read the input file and produce a list of error messages or a summary by department, if there are no errors.

2. INPUT. The input file contains records in the following format:

POSITIONS	CONTENTS
1-4	Company Name
5	Department Number
6-10	Paid Amount (up to 999.99)

3. ERROR LIST. Print a line for each record which contains an error as follows:

PRINT POSITIONS	CONTENTS
1-4	Company Name
6	Department Number
8-12	Paid Amount
14-80	Error Message

NOTE: Page and column headers are not necessary, but may be included if desired. Do not print valid records.

4. ERROR MESSAGES. The program should test each input record. If an error is found the record should be printed with the appropriate error message. It is only necessary that the program iden-tify the first error detected in an input record; it should then continue to test for errors in subsequent input records but abandon accumulating.

MESSAGE: COMPANY NOT IBNI
MEANING: The input record did not contain IBNI in positions 1-4.

MESSAGE: DEPARTMENT MUST BE NUMERIC AND > 0
MEANING: The input record did not contain a valid department in position 5.

MESSAGE: PAID AMOUNT NOT NUMERIC
MEANING: The input record did not contain valid numeric data in positions 6 thru 10.

5 DEPARTMENT SUMMARY. At end of job, if all of the records are valid and contain no errors, the program should print the total amount accumulated for each department and the total amount for all of the input records. Use subscripting to accumulate the department totals. Use subscripting to print department totals at end of job. Do not sort the input File and do not assume any pre-sorting. The total Paid Amount for all departments will always be less than one million dollars.

```
001000 IDENTIFICATION DIVISION.
001100 PROGRAM-ID. SAMPLE.
001200 ENVIRONMENT DIVISION.
001300 CONFIGURATION SECTION.
001400 SOURCE-COMPUTER. HP3000III.
001500 OBJECT-COMPUTER. HP3000III.
001600 SPECIAL-NAMES.  C01 IS FIRST-LINE.
001700 INPUT-OUTPUT SECTION.
001800 FILE-CONTROL.
001900     SELECT MEMOS ASSIGN TO 'MEMO'.
002000 DATA DIVISION.
002100 FILE SECTION.
002200 FD  MEMOS
002300     RECORD CONTAINS 132 CHARACTERS
002400     LABEL RECORDS OMITTED
002500     DATA RECORD IS MEMO-LINE.
002600 01  MEMO-LINE                  PIC X(132).
002700 WORKING-STORAGE SECTION.
002800 01  MEMO-ADD1
002900     03 FILLER                  PIC XXX VALUE SPACES.
003000     03 MA1-NAME                PIC X(30).
003100     03 FILLER                  PIC X(97) VALUE SPACES.
003200 01  MEMO-ADD2.
003300     03 FILLER                  PIC X(9) VALUE '   ATTN:
003400     03 MA2-EMP                 PIC X(36).
003500     03 FILLER                  PIC X(88) VALUE SPACES.
003600 01  MEMO-TEXT.
003700     03 MT-NO                       PIC 99.
003800     03 FILLER                      PIC XXX VALUE SPACES.
003900     03 MT-TEXT                     PIC X(72).
004000     03 FILLER                      PIC X(55) VALUE SPACES.
004100 01  TEXT-TABLE.
004200     03 TT-LINE  OCCURS 9 TIMES PIC X(72).
004300 PROCEDURE DIVISION.
004400 0000-START SECTION 01.
004500 0000-OPEN.
004600     OPEN OUTPUT MEMOS.
004700 0003-ALIGN.
004800     MOVE SPACES TO MEMO-ADD1 MEMO-TEXT MEMO-ADD2.
004900     MOVE 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' TO MA1-NAME.
005000     MOVE 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' TO MA2-EMP.
005100     MOVE ALL 'X' TO MT-TEXT.
005200 A1. WRITE MEMO-LINE FROM MEMO-ADD1 AFTER FIRST-LINE.
005300     WRITE MEMO-LINE FROM MEMO-ADD2 AFTER 1.
005400     MOVE SPACES TO MEMO-LINE. WRITE MEMO-LINE AFTER 1.
005500 0003A-TEXT-X.
005600     WRITE MEMO-LINE FROM MEMO-TEXT AFTER 1.
005700 A2. PERFORM 0003A-TEXT-X  8 TIMES.
005800     CLOSE MEMOS.
005900     STOP RUN.
```

ALTERNATE TEST

PROGRAM REQUIREMENTS

1. GENERAL. Read the input sequential file and produce a printed report that includes one detail line of print for every input record, a page header line on each page and a record count at the end of the report.

2. INPUT. This sequential file contains records in the following format:

Positions	Contents
1-17	City name
18-19	State abbreviation
20-24	ZIP Code

3. PROCESSING. Each input record will be counted and printed on the report. Before printing, the three fields must be formatted so that a comma follows the last non-blank character in the city name, followed by a blank, then the state abbreviation followed by another blank and the ZIP code. This will require a subscripted loop to move the state and ZIP one character at a time into the desired positions.

INPUT		OUTPUT
SAN FRANCISCO	CA93701	SAN FRANCISCO, CA 93701
NORFOLK	VA23151	NORFOLK, VA 23151
ANNANDALE	VA22003	ANNANDALE, VA 22003

4. OUTPUT. The printed report includes a page header line with the word PAGE in positions 1 to 4 and a sequential page number with leading zero suppression in positions 5 to 9. There should be a blank line after the header line and then up to 50 single spaced detail lines on a page. The city state and zip in the detail line should start in print position 10. The last detail line in the report should be followed by a double spaced line which contains a record count in print positions 1 thru 9 and the word RECORD in print positions 11 to 17. The record count should be printed with leading zero suppression and commas where appropriate. There will always be less than 5 million input records

FD, 17/2-83
RF/HH-4094L

IMPLEMENTING DISTRIBUTED APPLICATIONS IN A MIXED IBM-HP ENVIRONMENT

by Rolf Frydenberg
Project Manager
Fjerndata
Norway

CONTENTS

1. Abstract

This paper will present an overview of the basic methods of distributed applications. The advantages and disadvantages of the different methods or approaches will be discussed.

The different levels of sophistication at which distributed applications can exist, will be discussed.

We will discuss the available datacommunication software for accessing IBM mainframe from the HP3000, and what can be done by the user to supplement the inherent functions of those software products.

The realm of distributed databases will be presented, and we will look at the options in this area that are open to HP3000-users who have significant applications and databases on IBM mainframe.

2. Introduction

There are many approaches to implementing distributes applications. In this paper we will describe these according to what we have chosen to call their level of sophistication. We have found five basic levels of sophistication that are possible when application are distributed between HP3000s and IBM mainframes. Below is a presentation of these levels, as seen from the perspective of the HP3000 user.

A) Batch-only access

This is the most common situation, where a user submits a batch job through RJE or MRJE and retrieves output. The output may be directed to a printer or a file.

B) HP on-line, IBM batch

Many data-entry applications work in this manner. Transaction programs are run on the HP3000 that enter data into files. These files are transmittes to the mainframe as batch jobs, where they update a central (corporate) database.

C) HP Passthrough, IBM on-line

The HP3000 emulates an IBM3270-type cluster controller, and an attached terminal emulates an IBM3270-terminal, thereby allowing a user access to both IBM and HP3000 applications from the same terminal, but at different times.

D) HP batch, IBM on-line

A batch job on the HP3000 retrieves data (to a file or an application) by accessing on-line systems on the mainframe.

E) Simultaneous on-line

A user runs a program on the HP3000, that accesses the HP3000 or the mainframe, as the need arises. Data may be retrieved from, or entered into, HP3000 or mainframe-applications at will.

In most cases, level E would probably be acknowledged as the "ideal" that one should try to achieve. This is possible with DSN/DS between HP3000s. Of course, DS can also achieve the 4 less sophisticated levels as well. For access to IBM mainframes, levels A and B are the most common, using DSN/RJE or DSN/MRJE. The next three levels are possible by using DSN/IMF. Currently, most IMF-users only utilize level C - the PASSTHRU program in DSN/IMF or IMAS/3000.

3. Software for HP3000 to IBM communication

It is an established company policy of Hewlett Packard's to provide for three basic types of compatibility within its datacommunications strategy: Between HP computers, with internationally standardized networks and carriers, and to IBM mainframes, through IBM's System Network Architecture (SNA).

There are three HP datacommunication products for access to IBM mainframes: DSN/RJE DSN/MRJE and DSN/IMF. The common prefix for these product names - DSN - is an accronym for Distributed Systems Network. The product accronyms mean Remote Job Entry, Multileaving Remote Job Entry and Interactive Mainframe Facility, respectively. In this chapter we will look a little closer at these products.

DSN/RJE — Is HP's 2780/3780 emulator, for batch access to IBM mainframes and other types of computers. It is a one-file-at-a-time type of protocol.

DSN/MRJE — Is HP's Hasp Work Station emulator, providing more flexible and automated batch connection to IBM mainframes. It provides for transfer of multiple files, to mutiple devices, simultaneously.

DSN/IMF — Is HP's 3270 emulator, providing interactive ("passthrough") access to IBM mainframes, or program-to-program communication. It works with BSC or SDLC line protocol.

DSN/RJE and DSN/MRJE have been around for a number of years, and are much-used systems that well thought of by users. DSN/IMF is relatively new (announced as IML in 1980, and rewritten and renamed in 1981). So far, even thouhg the interest among users is high, the number of installations is still very low compared to MRJE and RJE.

RJE emulates a "dumb" communications processor, with up to 1 printer, 1 card punch and 1 card reader. RJE is an operator-oriented communications systems, i. e. an operator must issue commands - which files to transmit, where to route the received print or punch files, etc.. Only one user can use RJE at a time.

MRJE can emulate up to 7 printers, 7 card readers and 7 card punches simultaneously. MRJE does not require an operator for normal operation. Any user may transmit files through the MRJE user interface program, and determine the output-device or output-file for his own job. Print- and punch-output may be tagged by including a forms-code at the IBM host side, or by configuring each of the emulated devices to be equivalent with a specific logical device on the HP3000.

IMF is a multi-component software product. Seen from the user's point of view, it consists of two basic components: The Intrinsics (for programmatic access) and PASSTHRU (for emulation of IBM327X display terminals and IBM328X printing terminals). Other basic components are the communications driver, the cluster controller emulator, and the management program.

4. Distributed Databases

Distributed Databases has been a catchword in the DP area for å number of years. But what do we really mean when we use the word, and what are the real options and possibilities? In this chapter we will look at some of the possibilities, from an HP3000 user's point of view. This is not intended as a presentation of the "State of the Art" in distributed databases, but a summary of some of the factors that relate to distributed databases where both HP3000s and IBM mainframes are involved.

Let us start by defining what we mean by a distributed database:

> " A distributed Database is a database where the information that the database consists of, resides within more than one computer system. "

Any database - distributed or otherwise - generally contains two types of data: Catalog or pointer information (i.e. references to where data is located), and the actual data. A distributed database has at least one of these information types within more than one computer system.

Distributed systems that encompass IBM mainframes, or other mainframes for that matter, are almost invariably centralized systems. The mainframe is the "master", and the other computer, e.g. HP3000s, are "slaves" of the central mainframe. In Figure 4.1., below, we have illustrated the general organization of such systems, and named some of the software that is important when implementing distributed databases in a mixed HP-IBM environment.

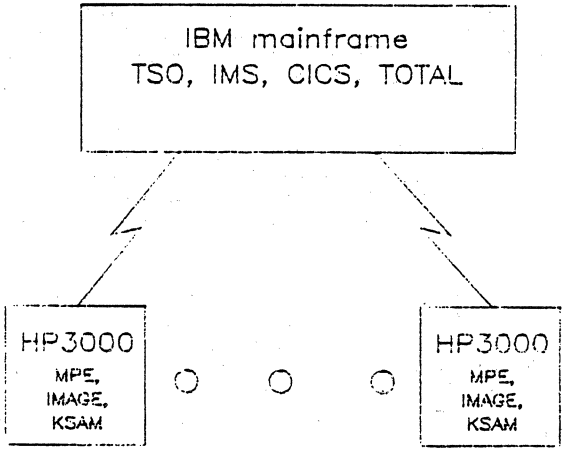


Figure 4.1: The Components of a Distributed System

There are two main approaches to designing a distributed database: Either separating data, or replicating it. In Figure 4.2 (next page), we have illustrated the "separate" approach, where each remote location (HP3000) has its own part of the database locally, and access the rest of the databse through the MASTER copy on the mainframe. Typically, the master is updated in batch mode every night, while the local parts of the database are are kept current continously.

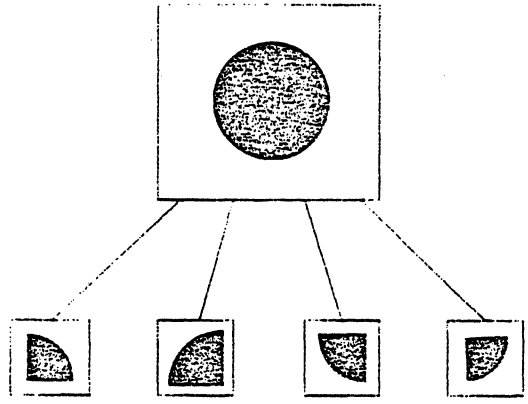


Figure 4.2: The Distributed Approach

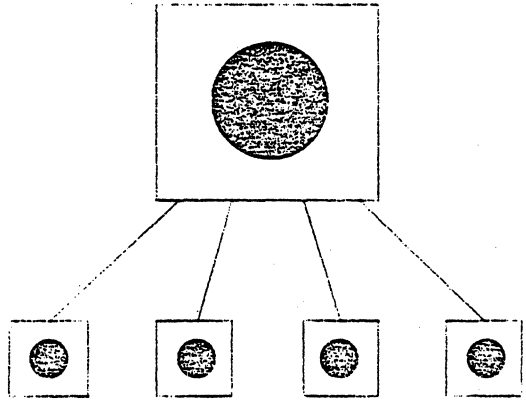


Figure 4.3: the Replicated Approach

Another approach is the replicated one: All the data is contained in local copies on all the interconnected systems. This is illustrated in Figure 4.3. This approach will minimize access time (only accesses to "old" data need be referred to the mainframe), but has several weaknesses as well:

Since all databases should contain the same data, a lot of backgroun activity must be performed (ideally without the users knowing it) in order to keep the databases concurrent.

The amount of data stored at the remote sites is greater than in most other cases, so the amount of local disk storage required is much greater.

The third, and easiest approach, is the one where only the catalog type information is distributed, but all the data kept at the central site. In Figure 4.4, below, we have illustrated this. Doing this will lead to a large number of accesses to the mainframe, but it is a "clean" solution, without the concurrency problems otherwise associated with distributed databases.

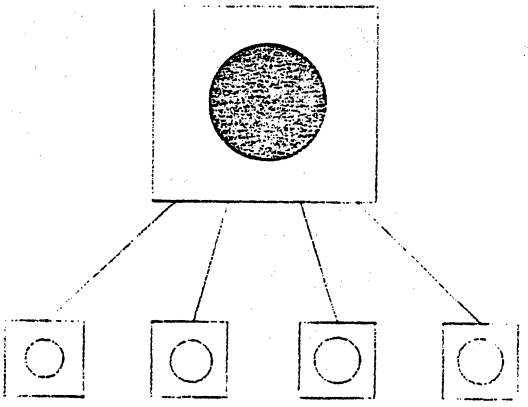


Figure 4.4: Distributed Catalog, Centralized Data

5. Remote data access - in batch mode

Batch mode access to remote data means that the data access seen from the HP3000 occurs in batch mode. This can be done through batch communications systems such as DSN/RJE or DSN/MRJE, or by accessing the mainframe as an interactive user, but from a batch job running on the HP3000.

The batch communications protocols for access to IBM mainframes, 2780/3780 and HASP Work Station, are primarily oriented to submitting a deck of punch-cards, in the form of a "JOB" to the mainframe, and retrieving output to a card punch or a line printer. The main difference between a 2780/3780 an HASP Work Station is the number of devices (card readers, card punches and line printers) that are supported.

Normally, RJE (the 2780/3780 emulator on the HP3000) and MRJE (the HASP Work Station emulator) are not thought of as components in a distributed application. But there are a number of options open to users in order to put more intelligence into the usage of these products. In Example A, below, we will look at one approach to integrating MRJE into the distributed applications of a user

Example A: Spoolfile Distribution.

The problem: Spoolfiles are received at the HP3000 from the centrally located IBM mainframe. Many of the files received are not supposed to be printed here, but at remote spooled printers.

The solution: This problem can be solved by comparing the IBM job-name and FORMS-code against a table , and then redirecting the spoolfiles according to the entries in this table.

This paper is intended to present implementations, so let us look at how we implemented this solution. First, how the operators "fixed" the problem using standard commands:

All the remote spooled printers - at this site - have a unique device class name, generally with the form LPxx, where xx is the logical device number of the printer. The mainframe users were instructed to use this device name as the IBM FORMS-code for the listings that they wanted to have transferred to their remote spooled printers.

Every time a forms request from MRJE with FORMS-code LPxx appeared at the console, the operator refused printing the spoolfile (REPLY pin,N) and redirected it to the remote spooled printer LPxx (ALTSPOOLFILE #Onnn, DEV=LPxx).

This of course, works. But the solution soon became a new problem: The users found that having their spoolfiles printed at their own printers was very nice, and so they generated more spoolfiles for their own printers, and more and more, and ..... Very soon the operators had little else to do than redirect spoolfiles. That was when we were asked to look at the problem.

The answer we found was quite straight-forward: Just implement an "automatic operator". Our implementation is the following:

A batch program runs SPOOK (HP's standard spoolfile utility program) as a SON process passing data through message-files. The following commands are sent by the program to SPOOK:

SHOW To retrieve status information on the spoolfiles, and to find out which files are candidates for redirection.

TEXT & LIST To get hold of the IBM FORMS-code.

ALTER To change the device class for the "current" spoolfile.

The program decides where to send a spoolfile based on two criteria - the IBM jobname (which by MRJE is converted to a filename) and the IBM FORMS-code. This information is contained in a file which is read when the program starts execution, usually at system start-up time.

Example B: A distributed memo-switching system

The problem: The user has message switching systems on both mainframes and HP3000s, e.g. MEMO on the IBM mainframe, JENNY or HPMAIL on HP3000.

The solution: A batch job accesses the mainframe, searches for memos to HP3000 users, copies these to local files, and then places them into JENNY/HPMAIL, and vice-versa.

With the advent of the automated office, electronic message switching is becoming increasingly more popular. On the HP3000 serveral systems are already on the market. These include HP's HPMAIL and Infomedia's JENNY. A large number of message switching systems is available on IBM mainframes as well.

Users with both IBM mainframes and HP3000s have several options available for integrating mainframe and HP3000 message switching systems:

Users can access the mainframe via the pass-through mode in DSN/IMF or IMAS, and read and create messages to other mainframe users on-line.

If the mainframe message switching system has a batch access facility, a JOB may be submitted regularly from the HP3000 (through RJE or MRJE) that can enter messages into the mainframe system and retrieve messages from it.

A special program may be written to access the message switching system on both computers, this program will then transfer messages between the systems.

For our own implementation, we selected the last of these possibilities. This is the most flexible one, and lets every user use the system with which he is most familiar, whether that is te mainframe or the HP3000.

This solution would have been very difficult to implement using standard DSN/IMF intrinsics, we estimate approximately 3000 - 4000 statements. By using the Automatic Dialogue Facility in IMAS, we could get away with about 200 Autodialogue statements for the IBM access part of it. The rest is done through TDP USE-files and standard JENNY/HPMAIL commands.

IMAS Autodialogues is a very high level special purpose programming language for accessing IBM mainframes. Autodialogues are interpreted by IMAS 3000, running interactively or in batch mode. for more information on what Autodialogues are, consult the Reference Manual for Fjerndata's IMAS/3000 system.

6. Remote data access - interactively

Data on the remote mainframe may be accessed interactively in two "modes": passthrough mode, where the user uses his HP-terminal to emulate an IBM3270-terminal; or in programmatic mode, where the user controls data transfer interactively, by using functions such as IMAS automatic dialogues.

Passthrough mode

In passthrough-mode, the HP3000 emulates an IBM3270 cluster controller, and HP-terminals are used to emulate 3270-type screen terminals or 3280-type printing terminals.

Before we go on to describe the two programs that can do passthrough-mode interactive access, let us look at the reasons why users want this type of capability.

One reason, of course, is economical: If you have on-line applications on the IBM mainframe as well as on the HP3000, and the same users need to access applications on both computers, you do not want them to require two terminals to do this.

Another reason, is standardization: If you want to have a single supplier of terminals, then you need passthrough-mode on the HP3000, since IBM does not supply passthrough-mode access to HP3000s. (But you could ask for it, of course.)

A third reason, is the datacommunications network: Asynchronous HP-terminals cannot easily share a datacommunications line with synchronous IBM-terminals. For screen terminals, DSN/MTS (HP's multipoint BSC protocol) might be used, but it is a costly solution, and is not compatible with IBM's SDLC protocol.

For one or several of the above reasons, and possibly some others as well, you have reached the conclusion that you want passthrough-mode interactive access to IBM mainframes from your HP3000s. The next question is: What are the alternatives?

Well, there are two alternatives: DSN/IMF comes complete with a program called PASSTHRU, which can emulate IBM3270 and 3280 terminals, and Fjerndata has a product called IMAS, which uses DSN/IMF, and can emulate IBM3270 terminals.

Since the main area of overlap between PASSTHRU and IMAS is in passthrough-mode emulation of IBM3270-terminals, let us briefly look at the differences, and how they affect performance and end-user productivity.

Programmatic mode

In programmatic mode, applications, files and databases on the IBM mainframe are accessed through programs. Here, as in pass-through mode, are two alternatives:

> DSN/IMF includes a set of INTRINSICs that allow programs written in most programming languages (from SPL to BASIC) to access the screen or printer images received from the mainframe.
>
> IMAS has a very high level language built into it, called the IMAS Automatic Dialogue Facility. Using IMAS Autodialogues screeen images are more easily accessed than through the DSN/IMF intrinsics, and the user has control while the autodialogue executes.

Using the DSN/IMF Intrinsics is not as simple as most other types of programming. Just a simple application that logs on to the IBM mainframe executes one command and displays the received screen on a terminal would probably require something of the order of 1000 statements. A "real" application needs a lot more, including procedures for error-handling and other utility type functions. IMAS, which is a program that uses the DSN/IMF intrinsics, consists of more than 10.000 statements.

Using IMAS Autodialogues, though, is quite simple. An autodialogue that logs the user on to an application, going through 3-5 screens to do so, typically does not require more than about 50 Autodialogue statements - including error-handling.

Autodialogues are used for a variety of other purposes than logon/logoff. The distributed message switching system described in the preceding chapter was implemented using the IMAS Autodialogue facility, using about 200 statements for logon, logoff and file transfers.

FUNCTION	PASSTHRU	IMAS
Terminal handling	BLOCK MODE	CHARACTER MODE, mod. fields
Function keys	8 Hard-coded function keys	4 userdefinable keysets
Terminals handled	HP Block mode terminals only	All HP terminals
DSN/MTS support	YES - block mode	Only through cluster contr.
Printer support	External print-device only	Internal printer supported
Access to commands and programs	Must EXIT	MPE MODE
Advanced functions	none	Automatic dialogues
Batch access	Only through INTRINSICS	Automatic dialogues

Figure 6.1: A Summary of differences between PASSTHRU and IMAS

7. Simultaneous access to IBM and HP

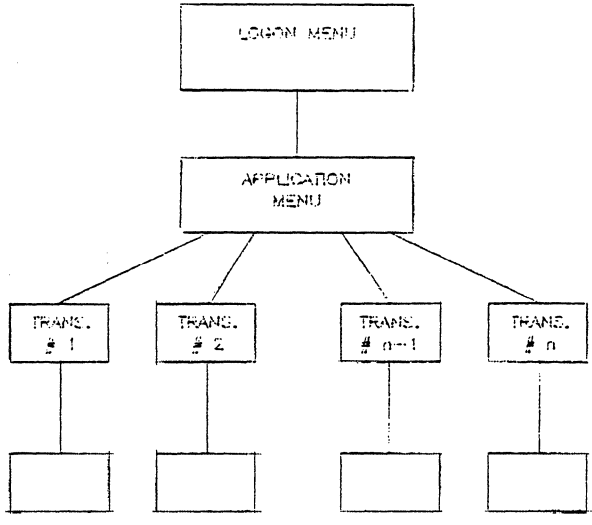
This is distributed databases in practice, where the programs access the data wherever it is - whether that is on the local HP3000, a remote HP3000 connected through DSN/DS, or on an IBM mainframe connected via DSN/IMF. This is the type of application that DSN/IMF was developed to support.

From the HP3000, one does not have direct access to the mainframe Data Base Management System. You cannot execute DL/1 calls directly but must view the mainframe database through an on-line application on the mainframe. The sets of screens that the mainframe application exists of may be viewed as a database, of the "hierarchical" type. In figure 7.1, below, we have illustrated this.

Figure 7.1: A "Database of Screens"

This example illustrates a typical mainframe application: First, you must logon to the application of your choice, then you select the transaction that you want to use, and finally you use this transaction, which may consist of several screens, to enter or retrieve data.

Comparing figure 7.1 to a database, the log-on menu is used to issue a request for a specific database, the "application menu" is used to select a DATASET and the transactions are used for GET, PUT, UPDATE, DELETE operations against the entries in the dataset.



8. Conclusions

What are the obvious conclusions that we can draw from the information presented in this paper? In my opinion the two most obvious ones are the following:

1. Distributed applications encompass a number of aspects apart from the most obvious one: Distributed Databases. In most cases, it isn't a distributed database the user requires, but the ability to access data on both IBM mainframes and HP3000s from the same terminal.

2. The best approach to implementing distributed systems is not necessarily through DSN/IMF, but rather through other software, e.g. IMAS Autodialogues or by writing programs that enhance other HP software products.

MODULAR CONSTRUCTION OF ON-LINE TRANSACTION SYSTEMS

THOMAS M. GAFFEY
FOREST COMPUTER, INCORPORATED
P.O. Box 1Ø1Ø
East Lansing, Michigan 48823

I. INTRODUCTIONS, DEFINITIONS AND DILEMMAS

If you talk to any HP Sales Representative or read any HP promotional information, it is likely you'll quickly come across a reference to the HP3ØØØ as a transaction processing system. It is certainly true that HP furnishes several valuable software tools that support a transaction processing system. As is well known, the IMAGE Database Management System has gained extremely wide acceptance and utilization from the HP3ØØØ user base. Other HP offerings like V/3ØØØ have made it possible to develop friendly, interactive, online application systems on the HP3ØØØ. MPE, however, is a time-sharing operating system. Under MPE, a program runs as a process; a process is a unique execution of a program by a user. The process consists of a data stack and code segments. Although the HP3ØØØ supports re-entrant code (code segments can be shared by all processes which are executing the same program), a characteristic of the MPE time-sharing environment is that the addition of users and their associated processes places a heavy increase in demands upon system resources.

In a time-sharing environment like MPE the user, session, and associated process are the logical unit of focus. The user has a great deal of flexibility in what he can do with the system. He can edit source code, compile a program, or execute a business application program. The operating system's job is to manage that user's interaction with the system. As multiple users begin to simultaneously execute the same programs, multiple processes which accomplish the same ultimate function are created. Each process consumes table entries dependent upon the number of data bases opened and many other factors. The cumulative effect is that an excessive amount of system resources become consumed.

The previous points are made to add credence to the contention that HP's use of transaction processing when applied to the HP3000 is too specialized a use of the term. Transaction processing systems in the real world are involved in the conducting of discrete business transactions in real time. Booking a hotel room or making a cash withdrawal from an ATM are two examples of transactions. In a true transaction processing environment, the transaction is the logical unit of focus. The capabilities granted a user are clearly defined and limited to a specific set of transactions. The addition of users to a transaction processing system should not impose the demands on system resources that the addition of users in a time-sharing environment does. Rather, system resources should be utilized only in proportion to the additional transaction volume which is generated.

The traditional approach to transaction processing on the HP3000 is to utilize menu programs which lead users to various application programs. The menu program leads the user to the initiation of a specific application program where the business transaction is conducted. When the transaction has been concluded, the user is returned to the menu for subsequent action. The overhead of process initiation on the HP3000 is significant, and where applications use this menu-to-program approach, the propagation times involved in the initiation of the individual processes becomes excessive. Also significant is the number of extra data segments, buffers, and various system table entries which are generated by each process. Many users of HP3000 who have watched their systems grow, know that in real life the number of users who can actually utilize the resources of the system is significantly lower than the physical number of terminals which can be attached to the system. These users have seen the downturn in the performance curve with the growth of their systems. Many users have had to lower their session limit because the average mix of active jobs caused system table maximums to be exceeded.

II. TRANSACTION SOFTWARE METHODOLOGY.

In order to design a more efficient transaction processing system on the HP3000, a different software design methodology must be employed. We have given this design philosophy and the software products that incorporate it the name ROUNDHOUSE©. Under ROUNDHOUSE, similar transactions are grouped into large programs which are known as "application units." The application unit runs as a conventional MPE process with it's own data stack.

Transactions enter the system from user terminals via small terminal driver programs which contain only those data areas needed for passing information to and from the terminal. The transactions are passed into the ROUNDHOUSE hub where they are checked for validity. ROUNDHOUSE associates a transaction with a process, and all executions of that transaction are routed to the same process. During times when large volumes of transaction activity is occuring, a transaction message unit (TMU) is queued in memory until the application unit can process it. Under extremely high volume transaction activity, these TMUs may be queued to disk. The overall effect of ROUNDHOUSE is to reduce the overhead associated with adding users to the terminal network while providing the user with quick access to any transaction on the system. With ROUNDHOUSE, terminal networks up to the limits imposed by hardware and system software can be realistically supported on the HP3000.

The basic differences between a standard implementation and a ROUNDHOUSE implementation are depicted in figure A. The standard implementation shows User "A" utilizing Process "A". When User "A" needs to access a transaction in Process "B", Process "A" must be terminated and a second copy of Process "B" initiated with time and system resources consumed. Each additional running version of a process under the standard implementation will duplicate ALL data areas of the original process (including Image data buffers).

Under the ROUNDHOUSE implemenation, User "A" enters a transaction for any application in Process "A", "B", or "C". No additional system resources are required or time consumed. The only increase in system utilization comes with transaction volume.

The most obvious advantage to the ROUNDHOUSE approach is the ease with which a user can access any transaction on the system. The actual manner in which transactions are grouped into processes is transparent to the user (as it should be).

Another advantage to the ROUNDHOUSE design is seperating the distinct processes of terminal I-O and transaction routing from the application itself. Each terminal driver could be tailored to a different type of terminal or setup to handle a different forms package or technique (thus affording a high level of device independence). Centralized transaction routing, in addition to giving the terminal user great flexibility, allows the addition of such things as transaction logging and recovery.

A third advantage is the decrease in system resources required to handle the transaction load. No resources are devoted to process initiation and a minimum of memory is consumed. A comparative graph of estimated memory consumed under a standard installation and a ROUNDHOUSE installation is shown in Figure B. The graph assumes all terminals are running the same application. The application uses approximately 11 KWords of user memory, accesses three Image data bases and uses V/3000. As is shown, there can be a significant decrease in memory usage as the number of terminals increase.

III. TRANSACTION CODING TECHNIQUES.

Since the transaction is the logical unit of focus under ROUNDHOUSE, application code development is best approached at the transaction level. While all related transactions could be bound into one application progam, the programmer should approach each transaction as a seperate entity wherever possible. At the same time, for ease of coding and maintenance, each transaction should use the same structure and many of the same routines. In particular, routines that access the database management system and perform terminal IO functions should not be duplicated.

To accomplish this, Forest Computer uses a modular approach to Cobol program development pioneered by the Illinois Law Enforcement Commission. This approach uses a standard "template" program structure into which code modules are inserted to form complete programs with varying capabilities and uses. Each of these code modules is restricted to performing a specific function, such as accessing a particular data set or performing a transaction.

Some of the advantages to following this approach are:

- The programmer is encouraged to think in terms of transactions rather than programs.

- Programs are broken into manageable components for ease of development and maintenance. Each component contains all source code specific to it.

- A programming structure is enforced to standardize processing flow and greatly enhance the ability of programmers to understand and work with the code written by other programmers.

- Within COBOL programs, all interfaces with other software systems such as VIEW/3000 or IMAGE/3000 are kept in discrete modules. This means that such systems can be modified or replaced with relative ease should it become advantageous to do so.

- Maximum flexibility in configuring on-line transaction systems is achieved. Transactions are independently coded and tested. At compile time, transactions can be selectively grouped into one or several programs to achieve the optimum mix of user flexibility and maximum throughput.

III - A. The TEMPLATE PROGRAM.

The "template" program contains the framework of a COBOL program with key words defining areas that may be filled in from code modules. Since the source from the code modules is always inserted into the prescribed areas, commonality of programming technique is encouraged. In particular, processing control sections encourage a top-down programming technique.

The general catagories contained in the template program into which code modules may insert program source are:

Compiler Controls
File Selects
Data Division
Global Working-Storage variables
Transaction Shared Storage (re-used by every transaction)
Transaction Specific Storage (retained in memory)
V/3000 Variables
Image/3000 Variables
Control Processing section
Screen Processing section
Utility routine section
Transaction processing section
Initialization section
Termination section

III - B. CODE MODULES.

Each code module performs a specific function such as accessing a specific data set or performing a specific transaction. Each contains the necessary Cobol

Identification, Data, Working-Storage (W-S) and Procedure Division entries needed for the module in one source file (by-passing the need to maintain libraries and keeping all related source in one place).

Some of the standard code modules which would come together to form a complete transaction program are depicted in Figure C. In brief, a description of these modules are:

Transaction modules - Each of these modules will contain the necessary code to perform one transaction. Kept in discrete modules in this manner, a transaction can be compiled as a single application unit or combined with other transaction modules to form larger programs. The basic categories found in each Transaction module are:

- Transaction permanent W-S definitions
- Transaction Shared W-S definitions (re-used each transaction)
- Screen layouts
- Edit control processing
- Update control processing
- Transaction edits
- Transaction updates

Utility modules - These modules can be used to insert commonly used routines into programs which need them. On the HP3000, these represent an alternate approach to using RLs.

Control modules - These modules are selected at compile time to allow the programmer to easily chose options such as bounds-checking or map listings.

For Data Base Management:

Three levels of modules exist to define access to an Image data set. All three modules are used when accessing the data set. They are, of course, defined only once and then used by all programs. Although not described here, similiar modules could be set up for KSAM or MPE files also. The data base modules in use at Forest consist of:

IMAGE module - this would contain the W-S global variables used by the actual IMAGE procedure division calls for each function (Open, Close, Find, Get, Put, Update, Delete, Lock and Unlock). Only one of these modules is needed per program.

Data Base module - There will be one of these modules for each Data Base accessed containing the necessary W-S definitions to utilize the IMAGE global routines for a specific Data Base. The open and close routines for the Data Base are included here.

Data Set module - There will be one of these modules for each data set used. It contains W-S definitions to utilize the IMAGE global routines for a specific set, a record layout for the set, and processing routines to perform all IMAGE functions for the set.

IV. CONCLUSION.

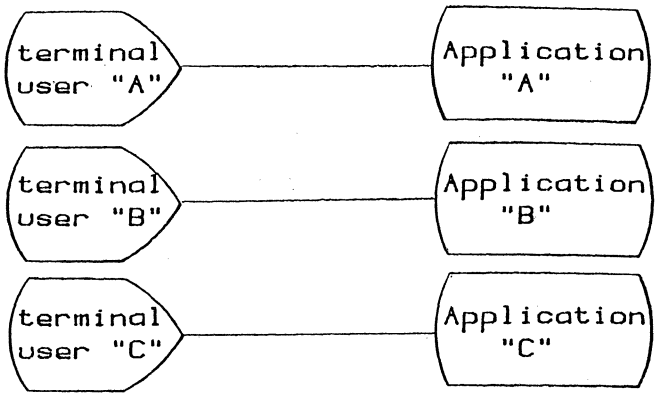
The software design techniques outlined work to circumvent the limitations imposed on transaction systems implemented on the HP3000. They make possible superior performance and the ability to increase the number of terminals using a system. These techniques allow use of the popular HP software offerings such as Image and V/3000 but are not restricted to systems designed around these products. Terminals using a transaction system following this design will have full access to all other MPE facilities and products such as text editors.

Programming for the transaction system, as would seem natural, is based on the transaction itself rather than on a larger program. All program source, in fact, is maintained in discrete modules performing a specific task - encouraging structured programming and easing program maintenance.

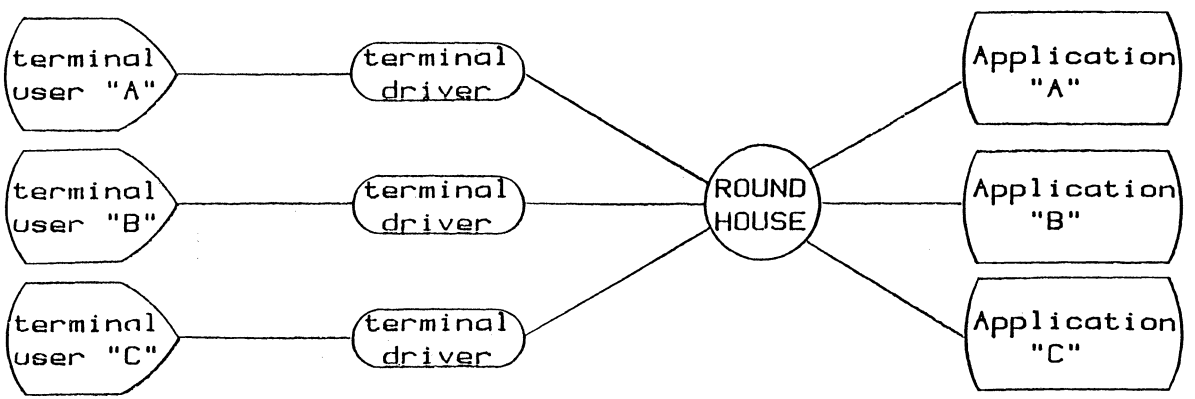
Finally, following the design techniques outlined here should allow for maximum flexibility in adjusting to future high-volume transaction loads. By establishing the terminal driver as a seperate process communicating with a transaction monitor, the road is paved to eliminate session mode in high volume activity or to off-load data communications overhead to a discrete processor. This could become important as the HP3000 provides no facility for support of custom data communications protocols.

Forest Computer is heavily involved in high-performance transaction systems used by a variety of companies, diverse in both size and specialty. We have found the ROUNDHOUSE methods efficient and practical, allowing transaction applications to be developed more quickly and at a lower cost than would otherwise be possible.

Anyone having questions who is unable to attend the presentation of this paper may contact Mr. Gaffey at telephone number (517) 332-7776.



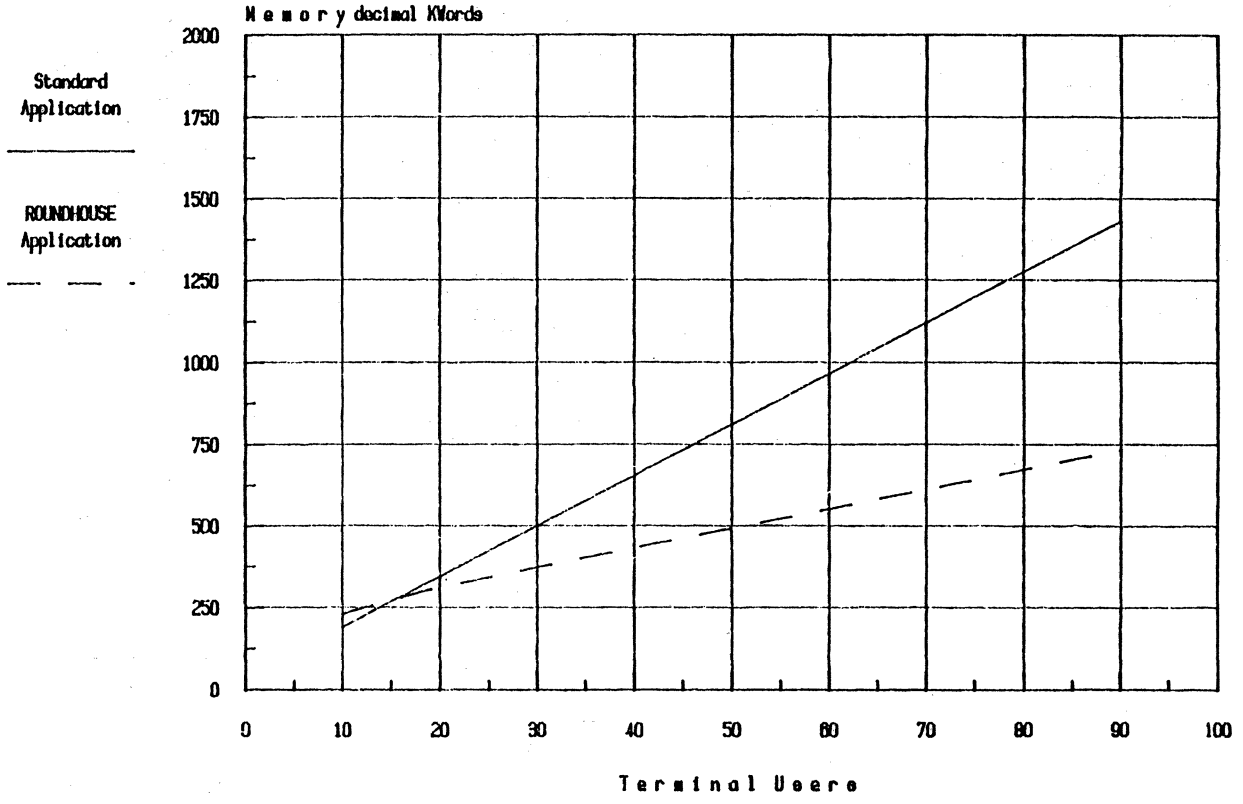
STANDARD IMPLEMENTATION

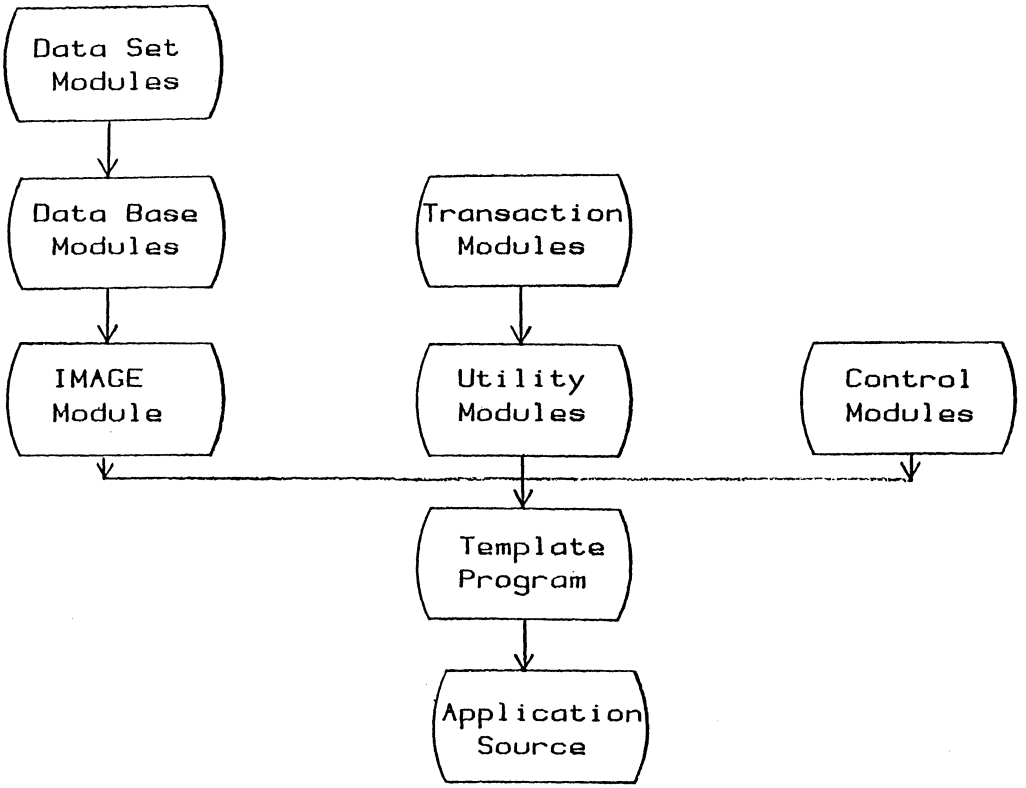


ROUNDHOUSE IMPLEMENATATION - FIGURE A

Figure B HP3000 Est. Memory Requirements

for average application





COMPONENTS OF TRANSACTION SYSTEM - figure C

The Future of Systems Development
or
Programmers are Users Too

by
Robert B. Garvey & Robert L. Womack
Witan Inc.
645 West 62nd Street
Kansas City, Missouri

Most of the presentations you will hear this week are technical. Our presentation will not deal with many tangible technical considerations but we trust that it will help you in systems development thinking.

First of all I want to change our perspective in an effort to help us see what some future possibilities may be.

An architect and industrial designer named Charles Eames produced 1 minute film to help visualize the speed of light a few years back. I want to relate to you a few frames from this film, and ask that you let your imaginations run free of MPE and maximum IOs per second for a while. I promise I will bring you down to earth after this journey.

Our movement away from the earth is going to increase at the rate of 10 times the existing distance each second, orders of 10. Our starting point is a hair on a man's hand. One second and 10 mm later we see a finger. The third a hand. The fourth, a body lying face down. Next a body on some sand with some other people around. Next a beach covered with people from 100 meters in the air. Feel the acceleration. A kilometer in the air we see a city on the beach. Ten kilometers up we see part of a continent. Our next glimpse is the earth and we are traveling 90 kilometers per second or 300,000 kilometers per hour. In our next frame the earth is a dot and the Sun is passing by. The next the entire solar system is in view. Then the Milky Way Galaxy. The next gets a bit confusing since we just passed the speed of light. In our next second we exceed the speed of the Starship Enterprise and we are well on our way to some interesting systems.

While we are on our journey lets visit some out of the way place to see if we can meet some celestial friends. Time doesn't mean much up here so I am going to speek in the past tence. Our last stop was the planet Zuldu, you and I met a most interesting lady named Lansia. Zuldu doesn't have any silicone and the Zuldunes have not developed any materials to replace their slow computing devices, but they have developed an uncanny and highly refined way of analyzing, refining, catagorizing and simplyfing the way things are done.

We met up with this lass and after a few social ammenities she was into our minds and wanting to know all about what we did and how fast this computer was that we talked about.

Early in the morning after all our cells were drained completely, she began a most humiliating explaination of how what we do could be done by any moronic member of earths society. If they would do things according to methods which she described. All the things we now do as programmers could be done by the machine if the system she described could be implemented on the computer. The automation of automation,

She gave us the general design for a system to increase the speed at which systems could be designed and implemented. She accounted for all the safeguards and "on the fly" modifications that we told her were inevitable in the life of a system.

It struck her that a suprisingly small amount of the earth time that has been spent in the design of systems to run on these electricity eating monsters has been devoted to the automation of the automation process.

Lansia identified our frustrations as programmers as things that the CONTROL program she proposed could take care of handily. If all the systems are done under CONTROL then changes that we may make will not adversly affect other parts of any individual system. Program logic errors will be reduced because the universe of control variables will be more narrowly defined and the data more manageable.

The tedium of writing code, compiling, segmenting, adjusting data bases and running the resultant system only to find an error or deficiency are reduce to making changes under CONTROL and EXECuting the system. At any point you can stop the execution and return to a mode where you can further modify then EXECute. If the system encounters a hard error, stopping at that point will position you at the procedureal point where the problem is occuring. She obviously was impressed with our discussion about the BASIC interperter.

We do hope that there will be some work for us after any such monster is implemented because it looked like the users really were not going to need us anymore. Worry not that is what retraining is all about.

Some other features of the CONTROL system are: The instruction entry and editing process is not by itself in the cycle, everything that is typed in must fit in the CONTROL system architecture. The CONTROL system is constantly asking questions about the data and procedure names and the implications of of activity created by our instructions.

Her CONTROL system relied on the maintenence of a large amount of data on all the systems under control, but it is data that we rely upon now and simply have not maintained in a unified way. We have not used one methodology long enough to have the data mean the same thing for very long.

The CONTROL program is the entire user interface to the machine. It is used throughout the life cycle of systems starting with the Idea stage, proceeding through the analysis and general design stages to the operational stage not far behind.

We asked her about changes to the CONTROL system, and on this issue she had some trouble, it seems that the people of Zuldu do not change very much, if we could return some time and tell her how we change her control system she can probably come up with a better way to do it.

Since we had to move on we could not get any more details or even remember all that we were told so what we need to do now is get back home and try to piece this together.

The salient features of this system then are: An overall control program through which you get to all other system functions relating to user data, systems data and procedures. Ability to make changes that are immediatly accounted for with respect to existing procedures and data. A base of data which relates each and every part to all other parts. Editing, keeping, compiling, segmenting and running are reduced to interactive entry, modification and execution. Execute time debug and interpertation. And finally the idea of the power yet inflexibility of an architecture.

With these things in mind lets look at some detail design requirements to connect these ideas together: There needs to be strong data typing. If the system is to be able to accept a new data item in development it needs to know more detail about the data than the IMAGE type, length and count. i.e. MDY-Date, YJ-Date, Dollar-big, Dollar-small, Toggles, ect. Given these standard types the editing of incomming data and presentation of outgoing data can be handled automatically.

Calls to other "systems" or "families" like names or text handlers under the control program. This data base of information means that a lot more is required than a data dictionary. The control system relies upon an organized set of relations between all the components of the system. Thus a

component base is required which might include Communications (screens or prompt) sequences), Files (the data dictionary), Operations (the proceedures, including families of general procedures) and Outputs (Reports and interfaces to other systems);

Ready to Execute
The notion that the system can run interpertively needs to be refined so that the system will run efficiently. Here we looked at the feature of the HP/9000 to run interpertively the first time and compiled thereafter. We might considered a more simplistic approach where the systems were analyzed at off hours to determine if they needed to be compiled (had they run interpertively more than some some dynamic amount) and if so they would be translated into some standard language preferably a block structured language like Pascal or SPL.

There is still the problem of the architecture being rather static and having the tendency to "tie one in" to that architecture for the systems that were developed during its tenure.

Assuming this is a picture of the shape of things to come what can we do today to prepare ourselves for its inevitability?

Data Typing
Begin using standard data types, develop standards for how they are stored, how much space is left on a formatted screen for input and output, how much space on a report, what are the Query edits, what is the standard callable to move the data from a file or data base buffer to the output buffer (what is the data area definition for all of that type for the programming language that you use). What are permissible calculations that can be performed on the data. Develop standard conversion routines to operate on the data. For example standardize you dates if possible and package your conversion routines and only do it once. Chose the RL, SL or COPY/INCLUDE approach and stick to it.

All Under One.
Develop an information system network. Represent your organization as a simple hierarchy and identify the general data structures for the entire organization which may eventually be served by your computer system. Code the organizational and data structure breaks so that the same naming conventions and organization within the MPE accounting structure exist. Put the standard routines where they are available to all and depending on your "standard routine" approach make the systems completely compillable at any point in time.

Develop standard menu routines and use callable subprograms and "timeshare" the stack.

Use or develop a data dictionary. The HP dictionary schema is the de facto standard and should be used if there is not a commintment to any other. Refine this to include a good way to handle the cross referencing of procedures and systems.

Develop a clean numbering scheme to identify your position within the the entire system tree. I suggest alternating numbers and letters.

Do all your documentation in machine readable form. TDP and Galley are very nice for this. Users can understand English outlining and with the addition of the powerful control words "IF" and "WHILE" you can represent the only three control structures you will ever need.

If you want to commit to a fouth generation system today do it cautiously. Consider that you may not have a long term solution and conversion to a more standard system may be easier using more traditional approaches. Ask yourself how many TRANSACT programmers are going to graduate this year.

During the analysis and design stages stress more the idea of data structure. Represent where possible the structure of the reports, screens and file (existing and proposed) using only the three primitive structures.

Your design document should be the users manual, there should be no other general design document. Realize that most systems do not need a daily users manual. After brief training this manual will be used for reference only. Keep it in such a way that as changes are made in things like data bases and files that the reproduction of the manual will not be a process of updating the manual but simply running the job to produce the manual relying on the dictionary and forms file and standard "use of the system".

DS X.25 on Public Data Networks: performance and cost issues

Ted Rypma

Hewlett Packard (Canada) Limited
6877 Goreway Drive, Mississauga,
Ontario, CANADA L4V-1M8
(416) 678-9430

ABSTRACT

Description of an eight system HP3000 network implemented on the Canadian DATAPAC packet switching network. The Hewlett-Packard Systems Engineering Organization has a network connecting all demo HP3000's using 2400 bps DATAPAC access lines. The network implements HPMAIL and is used for internal mail and transferring patches and patch information. Network performance and cost issues are discussed.

Introduction

Hewlett-Packard (Canada) has eight sales offices which have at least one HP3000 computer system for sales use. These systems are normally used for demonstrating Hewlett-Packard software, conducting training courses, local document and graphics processing needs and verifying software bugs and problems.

At present, a need exists to be able to demonstrate networking and electronic mail capabilities and verify problems in both. In addition, a means of reducing telephone costs, decreasing time involved in contacting people in other offices and getting timely delivery of critical documents and software changes is always welcome.

All of the above objectives were met by installing DS/3000 with X.25 and DATAPAC 3000 access lines into all demo computer sites and running HPMAIL on these systems. Each system was then able to communicate directly with all other systems in sales offices equipped with HP3000's.

A pilot group of users in each sales office was initially trained to use HPMAIL in a short session and then allowed to transfer any messages and documents to other users in the pilot groups in the same or other offices. The remainder of this

paper discusses issues in the implementation of this HPMAIL network and experiences from its operation.

Network Description

The Hewlett-Packard offices linked by DATAPAC and DS/3000 are located in Dartmouth, Montreal, Ottawa, Toronto, Winnipeg, Calgary, Edmonton and Vancouver. Each office has one 2400 bit per second DATAPAC 3000 access line with eight switched virtual circuits. This makes it possible for any one system to be in simultaneous communication with all other systems and still have one free virtual circuit. Permanent virtual circuits were not used as they are inappropriate for the application and are not supported by DS/3000.

A relatively slow 2400 bps was chosen as a line speed primarily for the nature of the traffic on the network - a mail system does not require immediate interactive response. The volume of messages was felt to be such that full utilization of bandwidth available would only occur for perhaps five minutes at the hour and fifteen minutes at three specified times during the day or night. The cost of the network itself was thus kept to approximately the cost of 201C switched modem rental.

The maximum Level 2 (Frame) window of 7 was chosen to best utilize multiple circuits active on any one line (each DSLINE command to a different node requires a virtual circuit).

Network Performance

The timing of HPMAIL delivery is controllable to fifteen minute intervals, thereby allowing considerable control over network congestion and system overhead. At this time, all systems in our network transmit to another system at a particular time local to it: 6 am, 10 am and 3 pm. Urgent mail is sent on the hour from 6 am till 6 pm local time for the destination system. This scheme guarantees that mail will arrive at very specific and predictable times, allowing easy scheduling of system down times. It also guarantees, however, network congestion and numerous logons at those same very predictable times.

The mail network should, therefore, be set up to stagger delivery times, with routine mail being delivered at non-peak times if possible. This may of course be at odds with a desire for timely message delivery and a compromise may be necessary. Primary mail transmissions should be timed such that they avoid concurrency with many other systems sending to the same node. Urgent mail will normally be infrequent enough to cause few problems. As a result, there will be more available virtual circuits for network applications other than HPMAIL.

Analysis of message traffic showed DS/3000 overhead and DATAPAC delays adding 200 msec to the transit time for a packet. Total packet transit time was on average at 2400 bps 1.1 seconds. From this it is relatively easy to determine the time required

to transmit given quantities of mail to other systems, assuming no competing traffic and no other system activity.

Following are packet counts and timings determined from use of HPMAIL:

Activity	Packets	Time
Logon/Logoff	16	25 sec
HPMAIL setup	19	66 sec
Sector of data	1	1.1 sec
Acknowledgements	1 or 2	1 to 2 sec
Example TDP and short program file	82	140 sec

(309 records of 88 bytes and a 6 record program file.)

Sending even the smallest message will take a minimum of 30 to 35 packets plus 90 seconds time plus one Logon/Logoff. This leads to the same conclusion that can be drawn from most data transfers, whether to a disc drive or on a network: "Make each transfer count". For HPMAIL, this means allowing mail to accumulate and, if possible, sending one longer message instead of a number of short ones if network and system utilization are to be minimized.

A brief comparison with DSCOPY will serve to put HPMAIL performance into perspective. DSCOPY is at the moment the most efficient means of transferring files from system to system using DS/3000. It provides none of the features of HPMAIL except the ability to transfer disc files and has none of the overhead associated with routing and delivery.

Following are packet counts and timings for DSCOPY transfer of the same files as for HPMAIL above:

Activity	Packets	Time
Logon/Logoff	16	25 sec
DSCOPY setup overhead per DSCOPY execution	12	17 sec
Sector of data	1	1.1 sec
Example TDP file and short program file	88	110 sec

The logon and logoff overheads are the same and the data transfer time and number of packets for the TDP file were similar, but the program file required almost 50% more time and packets with DSCOPY. This is as a result of the number of FGETINFO and FFILEINFO calls made by DSCOPY to handle correct file creation and validation. DSMAIL does this later under local user control.

The comparison shows DSMAIL to be at least as efficient as DSCOPY,if not more so.

Cost Analysis

Line Costs

Each of the 8 computers in the network has a single 2400 bps DATAPAC 3000 access line with 8 switched virtual circuits. This configuration costs out as follows:

Item	Cost
2400 bps DP3000 access	$165 / month
7 extra SVC's à $3/month	$ 21 / month
Monthly line charge / node	$186
Monthly line charge, network	$1,488

Packet Costs

Packet charges are more difficult to estimate, as mail needs vary widely and are not very predictable. Some assumptions can be made to aid in calculation of packet costs:

The cost per kilopacket (kpac) varies from $0.50 to $1.50, depending on source and destination node. This will be averaged out to $1.00/kpac.

Routine mail delivery is done 3 times per day.

Urgent mail id sent on two occassions per node per day.

All 8 nodes are up all the time and able to send/receive mail.

No retransmissions are necessary due to network or system failures.

Call setup charges of $0.005 per call are ignored.

From these assumptions, the following minimum daily packet utilizations can be calculated:

	Packets	Cost
Per node routine mail sent (Logon/overhead/Logoff)	108	
Per node urgent mail sent	72	
Daily mail sent by one node to any one other node	180	
Daily mail sent by one node to all other nodes	1260	$1.26
Daily mail sent by all nodes to all other nodes.	10,080	$10.08

Assuming there are on average 30 days per month, this results in a total of approximately $300 per month in packet charges without sending any mail. The number of characters of mail for any system must be estimated and divided by 256 characters to estimate the packet charges for actual mail sent. The most significant fact for this calculation is the cost of 256,000 character of data: $1.00!

For approximately $1.00, a user can send 130 full screens of data from an 80 column by 24 line terminal. Even assuming there are 30 users per system and they send a full screen message to each other user on each other node (30*7*30 or 6300 pages), the cost would be $47.25. In actual fact, a user is not likely to send more than five to ten different messages to OTHER nodes per day. This results in 40 to 80 packets or $0.04 to $0.08 per day per user.

Summary and Conclusions

Use of an electronic mail system has reduced the time needed for members of the Canadian Systems Engineering Organization to communicate with one another throughout the country. Phone calls back and forth from office to office have been significantly reduced.

Costs for setting up a nation-wide packet-switched mail network are not very different from having one 2400 bit per second 201C switched modem per system. The use of DATAPAC with DS/3000 provides great flexibility and convenience in implementing HPMAIL - one communications controller per system is required to give concurrent access to all other systems in the HP demo system network.

Line and modem costs for an eight system HP3000 network are $1,488 per month.

Estimated packet charges for mail delivery three times per day and transmission of five full page messages per day for each of

30 people per node is $180 per month, assuming 20 working days in a month.

The MPE Memory Dump; or How to Make a Statue of an Elephant

Jason M. Goertz
Hewlett-Packard
Bellevue, Washington

Introduction

In the past several years, the number of HP3000 sites has increased in number dramatically. The 3000 has been called into service to perform more and more complex and demanding applications. Applications that use Privileged Mode, Process Handling, Message files and other advanced features and capabilities are becoming almost commonplace. Along with this increasing application sophistication there has developed, necessarily,. an increasingly sophisticated user community, who require more complex debugging aids and tools to facilitate development of these applications. It is for this class of user that this paper is written.

While developing this type of application, particularly ones using Privileged Mode, MPE integrity is sometimes compromised. Many times this results in some kind of system interruption, usually a system failure or hang. In almost all cases, in order to determine the cause of the problem, the MPE memory dump is the most concise, economical, and easy tool available. However, information on how to read and interpret the mountain of paper produced is virtually non-existent. Even within Hewlett-Packard, only recently has organized training on the subject been available.

This paper is an attempt to fill this information gap. Please note that MPE dump reading and (especially) interpretation requires in-depth knowledge of MPE and subsystem internals, not to mention a lot of practice and experience. This point cannot be stressed enough. This paper is not intended to replace any of these things, but to give a capsule summary of some of the more basic and important facts and methods

An Explanation

Before beginning, the title of this paper must be explained. In the two years, and particularly the last year, that I have been reading dumps on a regular basis, I have evolved an answer to the two questions most often asked by people in my office, namely "What do all those ones and zeros mean?" and "What do you look for?". (The answer to the third question, "Do you really enjoy doing that?" is worthy of another paper, or at least a few hours of discussion accompanied by several doses of liquid refreshment.) The dialog which ensues after either of the above queries is something like:

I: Do you remember elephant jokes?

They: Sure.

I: Remember the one about the statue?

They: No.

I: Well, its like this: How do you make a statue of an elephant?

They: I give. How?

I: You take a hammer, a chisel, and a block of marble, and you knock away everything that doesn't look like an elephant. Reading a dump is basically the same idea. You take the dump, a Tables Manual, and PMAP's, and you find the part of the dump that doesn't look like MPE. At that point, you've found the problem.

It's then that the poor sod who "had to ask" usually walks away shaking his head in bewilderment. With this thought in mind, let us proceed.

Fundamentals

Before starting to read and interpret a Memory dump, it is necessary to understand exactly what one is.

When the system stops, for whatever reason, the contents of memory are "frozen" at that instant. In addition, the microcode of the machine dumps the value of the CPU registers (DB, Q, S etc) into a special area of low memory. A serial medium, usually tape, is mounted and the contents of memory, starting at low addresses and proceeding through the highest word, are dumped serially to the medium. This is accomplished by either microcode (Series II/III, herewith referred to generically as SIO machines) or by software (SDFLOAD on Series 30/33/40/44/64 machines, herein

referred to generically as HP-IB machines). After the machine is brought up, a program called DPAN4 is run under MPE control that reads the tape and formats the contents in a meaningful form. The resulting listing is what is commonly referred to as "the dump".

It is important to realize what this listing represents. It is basically a "picture" of MPE in memory. In essence, it is MPE, as much as any physical thing can "be" software. In order to interpret this "picture" of MPE it is critical that the interrelationships of the various parts be understood. Therefore, the very first thing that must be acquired to read a dump is a thorough understanding of the workings of MPE. This is not possible to do in the scope of this paper, but a few key facts and concepts will be presented.

The memory of the HP3000 is divided into sections or "banks" of 64KW each. Banks of memory are treated equally within MPE, with one exception, and that is bank 0. This is where MPE (specifically INITIAL) places most of its critical system tables. The reason for this is that, originally, the HP3000 was a 64KW machine, and all of MPE and user code were in this memory. All of the memory resident (nonswappable) tables are in this bank, especially the Code Segment Table, Data Segment Table, Process Control Block, IO and Disc Request Queues, and Memory Allocation Manager (MAM) tables. A great deal of information can be gained from understanding just the first 5 above, plus the format of the user stack.

Code and data are separated in memory, and is accessed in variable length "chunks" called segments. It is necessary that MPE, as well as the hardware (microcode) keep track of where in memory or on disc these segments are located. The CST and DST are used for this purpose. The Code Segment Table is really divided into two parts, the CST and the XCST (Extended Code Segment Table). The latter was introduced in the Series II when the increased memory size necessitated a larger storage of Code segment information. Each entry is four words of memory, and contains information on location (either bank and offset or disc address), whether it is in memory or not, and its length. Other data is also stored, such as whether it is memory resident, or (in the case of a Data segment), whether or not the segment is a processes's stack.

The CST is used to point to Code segments that are resident in an SL file. Program file segments are kept in the XCST. For each process, there is a bank of XCST entries, each entry with the same format as the CST. CST's currently are numbered from 1-%277, and XCST's from %301 to %377. It is this numbering range that is used by the microcode to represent logically contiguous code space, as well as by

DPAN4 and the dump reader to determine the origin of the segment. These tables provide data to determine exactly what was executing during and prior to the failure.

The Process Control Block (PCB) is used by the dispatcher to keep track of the various processes on the machine, and which one will run (be dispatched) next. (A process is defined as a unique execution of a program at a point in time.) A process will always have at least one Code and Data segment, plus a PCB entry which ties the whole thing together. The PCB also contains extremely valuable information to the dump reader, such as why a process was waiting (and what event it was waiting for), as well as whether the process was attempting to abort, where the DB register was, etc.

The primary IO tables, the IOQ and the DRQ, are a list of those IO's that are waiting to occur or have just occurred. The structure of the two tables is almost identical, although there is a bit more information in the DRQ. In order to fully interpret the IOQ it is necessary to have a good understanding of ATTACHIO (the software interface to the IO system), and the individual device driver. However, these two tables can be invaluable to the dump reader who is facing the analysis of a system hang.

The data structure which gives the best "history" of what lead to a failure is the process's stack. The stack data area is delimited at various places by the CPU registers DL, DB, Q, S, and Z. Below DL is the PCBX (PCB Extension) which is used by MPE to store non-critical scheduling information and is not accessable to user code. This area contains some relavent data structures and information, most notably file control block pointers, as well as pointers to two other important process tables, the Job Directory Table and Job Information Table (JDT and JIT).

When the program executes, it issues PCAL instructions which cause control to be transferred to another procedure, most often to a system segment, such as IMAGE or the filesystem. The PCAL instruction, as part of its normal operation, places a four word marker on the stack (at the current S pointer). This marker contains data which allows the environment at the time of the call to be preserved so that a proper return can be executed. The data includes the current value of the X, P, and Status registers, as well as the number of words between that marker and the previous one. We can see that if we start at the topmost marker and work backward, we would have a history of what code the process had executed until the time of the failure (if the process in question was the cause of the failure), or what the process was doing before it gave up the CPU. This is called a Stack Marker Trace, and DPAN4 formats it twice, once by itself in the formatted portion, and again when the whole stack is formatted.

A similiar structure to the process stack is the Interrupt Control Stack. This is a stack that resides in low memory, and is used primarily by the IO drivers and the dispatcher. In the case of a Memory Manager or IO system failure, the ICS is examined the same way a normal stack is to determine what code was executed before the failure. Typically, if this data structure is involved in the examination of a dump, the problem is most likely an MPE problem, and therefore up to HP to analyze.

The formats for most of these tables can be found in the System Reference guide, in addition to a very detailed description of the interrelationships of the tables. The MPE tables manual (PN 32002-90003) provides a detailed description of the formats of the various tables, and a description of the various data elements stored in them.

Dump Format

The actual dump listing is divided roughly into two parts, commonly called the "formatted" portion and "unformatted" portion. In reality, both portions are formatted, the first part more elaborately and with more detail and intelligence than the second. The formatted portion consists of selected tables which DPAN4 prints with the various fields labeled. Additionally, most of the various fields are printed with mneumonics (such as C for Core-resident, or S for Stack). The unformatted portion is just an octal dump of memory, starting at bank 0. The various tables are labeled if DPAN4 can determine their identity. All tables used for memory managagement that are in memory, such as the region trailers and headers, are printed in such a manner as to allow the reader to separate them from the corresponding segment. For each segment, the left hand side has not only the bank relative address, but the segment relative address also. For most data segments, the right hand side has the ASCII equivalent of the contents printed, with periods representing nonprintable characters. We will now discuss how DPAN4 formats the various tables mentioned above.

The first page (Figure 1) is called the Register Page. This gives a listing of all the CPU registers at the time of the system halt. The stack registers are on the left, followed by the Code Segment registers. Next is the X, Current Instruction Register (CIR), and other registers that vary among the different hardware types. An interpretation of the various bits and fields in the Status Register formatted in the next column, followed by the other hardware dependent registers. While the latter is sometimes of interest, especially when diagnosing hardware problems, the first three are more commonly used. Below the box containing the registers are contents of low memory. These words of memory are used by the microcode to

mark the beginning of the various critical tables, such as the CST, DST, XCST, and the PCB. The ICS limits are stored here also.

One very useful piece of data stored in low memory is a pointer to the PCB entry for the currently running process. If this is nonzero, then a process was running, and is usually the process which caused the failure, although it is possible to have a current process and also have something, such as an IO driver, running on the ICS. Code running on the ICS is indicated by several things on the register page. On all machines except the Series 64, the DL register (far left) being set to -1 (%177777) indicates that the current stack being used was the ICS. On the Series 64, as well as the other machines, there is a bit in the CPX1 register which DPAN4 formats in the last column of the register box. DPAN4 labels this the ICS FLAG, and is either on or off.

Following this, the CST (Figure 2) is formatted. When DPAN4 runs, it interrogates the file LOADMAP.PUB.SYS to determine the names of the segments. These are printed out on the line, along with all of the other data from the CST. Next is the XCST (Figure 3), which is formatted by groups, each group representing the XCST entries for a particular process.

The DST (Figure 4) is listed in almost identical format to the CST, with the names of the various system tables being printed on the appropriate lines.

Next is the PCB. This is divided into two halves, as there is too much data in each entry to be formatted on one line. The first half of the PCB (Figure 5) shows the process tree information, the wake and event masks (used by the suspend and activate mechanism within MPE), plus the psuedo interrupts that the process has accumulated, such as from a break, control-y, or an :ABORTJOB executed on that process's job/session. The second half (Figure 6) has scheduling information, used primarily by the dispatcher, bits which show what resources (SIRs, SETCRITICAL) the process holds. In addition, various pointers and other data are formatted.

The IOQ and DRQ (Figures 7 and 8) are similiar in format. DPAN4 formats each in two parts, an "in-use" list and an "available" list. The inuse list for the DRQ is additionally divided into a list for each disc configured on the system. For the dump reader, the available list is a recent history of IO activity on the system, sometimes giving a clue to the cause of the failure, or at least to what the failing process was doing. The inuse list can give invaluable data as to what IO's were pending and why they had or had not completed, as well as the relative order in which they had been queued.

The data structure which DPAN4 does the most work on is probably the data stack. As DPAN4 is dumping main memory, (the "unformatted" portion of the dump), it checks each data segment that it encounters to see whether or not it is a data stack. If it is, it formats several pieces of data from the bottom of the stack, an area known as the PXGLOB area. This data is very useful to quickly identify several things about the process, such as what $STDIN/$STDLIST device was assigned, what Job/Session number was assigned, and what the JIT and JDT dst numbers are for that process. After this, the stack markers are repeated, and the PXGLOB, PXFIXED, and PXFILE areas are printed. (See Figure 9). DPAN4 delimits and labels the various file control blocks, in the PXFILE area. When a loction of memory is reached which is pointed to by a CPU register (for that process), DPAN4 prints a line of asterisks and labels this register. This is also done for each stack marker as it is encountered. Alongside the marker, the segment name is printed, just as it was in the stack marker trace, above.

Additionally, DPAN4 prints a "table of contents" at the beginning of each bank of memory. At the end of the dump, it produces a list of the main tables, and the page numbers on which that table appears in both the formatted and unformatted portions.

We now have at least a passing familiarity with the format of the dump and with the functions of the tables that the dump represents. Let us now discuss how to use this information.

System Interruptions

There are five types of system interruptions, and are defined as follows:

1. System Failure. This is caused when some code, usually MPE, detects some error condition, and calls a procedure called SUDDENDEATH. This procedure prints the all too familiar system failure banner, and halts the machine. These failures can be caused by a hardware problem which the software detects at a later time, a system table that has been altered in a way that causes integrity loss in MPE, or sometimes by an invalid parameter passed to a system primative.

2. System Hang. This is when the system is in a pause state, but no response can be obtained from terminals. Many times, the system will hang when users try to logon or logoff, or run a program. In the case of a hang, the hardware is running, but the software cannot run for some reason. It

is important that the exact symptoms of the hang be known. Without this knowledge, it is often difficult to know where to start looking in the dump for the cause.

3. System Loop. This occurs when a high priority process, such as a system process or datacomm monitor process (or user process in linear queue) gets into a tight loop, and does not allow another process to run. Another possible cause is when a process has PDISABLE'd (turned off process dispatching), and has not PENABLE'd properly.

4. Silent Halt. This occurs when the microcode detects an "impossible" condition, such as an ICS overflow. These types of halts are silent only on SIO and Series 30/33 machines. On SIO machines, this usually will cause the System Halt light to come on. Other HP-IB machines will print a HALT n message, where n is a number which indicates the type of halt encountered. Most often, this indicates a hardware problem.

5. Port lockout. A particular port will not respond. Usually, this is an application problem. Most often, this is associated with a process handling application, or a problem with a specific peripheral.

We will examine each of these dump types, and summarize what to look for in the dump.

Some tools

Before delving into the actual analysis of the dump, it is necessary to accumulate a few tools which make the dump reading process easy. Besides the dump, it is necessary to have the PMAPs of the various MPE modules, and a current MPE Tables Manual. The Tables manual can be ordered from HP, PN 32002-90003. A true PMAP listing of the MPE modules are only attainable by doing a PREP on the various MPE modules. Since this is not possible for most users, besides being very difficult and time consuming, an easier method is necessary. A program is available called SLPMAP, which reads an SL file (usually the system SL) and produces a PMAP-like listing for each segment of the SL, in alphabetical order. While the segment locations are not totally accurate, they are close enough to locate the procedure which was was executing from a stack marker trace. If a particular application is involved, especially one that utilizes Priviledged Mode, the source code and PMAPS for the code involved needs to be gathered as well.

Analysis

The first thing that must be done when analyzing a dump is to determine if the dump was even valid. Sometimes the contents of memory is so corrupt that DPAN4 cannot determine where certain tables are in memory. When this occurs, a diagnostic message is printed out, and a list of the exact tables that could not be formatted is given. DPAN4 will then say that it is suspending the formatted portion of the dump, and that memory will be printed in an unformatted manner. At this point, DPAN4 prints an octal dump of memory starting at bank 0 and proceeding to the end of the data that was on the tape. This dump is virtually worthless. It would be extremely tedious and time consuming to try to analyze this dump, and it is even a waste to print it.

The next thing that must be done is to determine what type of failure occurred. Typically, if a user is analyzing a dump that occurred on his or her system, then the type of failure will be painfully known. Assuming the type is not known, or the type is uncertain, the following analysis should be done:

1. Look at the current CST number, and determine if the segment HARDRES was executing. This can be determined by checking the formatted CST table or the file LOADMAP.PUB.SYS. If so, then SUDDENDEATH was probably called. The PMAP can be checked to confirm this. Then check the Current Process Pointer. If non-zero (and the other CPU registers do not indicate that some code was running on the ICS) then the failure was most likely caused by a particular process. If the registers indicate that something was on the ICS, then an IO driver or the dispatcher/Memory Manager probably caused the failure.

2. If the Current Process Pointer is non-zero and the current CST register is not HARDRES, or if it is and the current P register is not in SUDDENDEATH, then the dump is probably a system loop. Information from the site is invaluable in this case.

3. If there is no current process and the CIR register contains the PAUS instruction, then the dump is either a hang or a port lockout. Information from the site makes the analysis much easier for this type of dump.

4. If the CIR register contains an instruction other than a HALT or PAUS, then this is most likely a silent halt. It is very difficult, however, to

tell this dump from a system loop. Halts are usually caused by a few machine instructions, and analysis of the specific instruction is necessary in this case. Site specific information, such as configuration, is usually necessary to analyze this type of failure.

Determining whether the problem is hardware or software is sometimes very difficult, since hardware problems can often manifest themselves as software. There are a few failures which are readily identified as hardware just from the description. For example, SF15 is typically caused by a nonresponding hardware module, such as memory controller or GIC on HP-IB machines. SF201 is the same thing on SIO machines. Many types of hangs are caused by the disc subsystem or perhaps by communications boards. In any case, there are no hard and fast rules for determining whether something is hardware or software. The common types of hardware failures will be mentioned in the following discussion.

Figures 1 and 10-12 show an example of the first type of failure. The System Failure number is 311, which indicates process aborting while critical. In this case, the program is one called BADLABEL. The source code was modified to produce this failure. Figure 1 is the register page from the corresponding dump. Notice that the current process pointer is non-zero, and the DST registers are pointing to a stack, and that the DL register is not -1 (%177777). All of these things point to the fact that a specific process was the cause of the failure. Figure 10 shows the stack markers from what DPAN4 found to be the current process. If the process's name was not known, a simple technique can be used to find the name. The PCB entry is found, and the father's pin number determined. Assuming the father is a UMAIN (Command Interpreter) process, and that its stack is in memory, the :RUN or subsystem command can be found at DB+1 in that stack. We see from the current stack markers that the last user segment to execute was %301 at address %1510. Looking at the PMAP for this "application" (Figure 11) we see that the procedure that was executing was PROCESS'ENTRY. Subtracting the address of %1510 from the starting address in the PMAP of %1024 we get a net result of %464 which points to the actual line of code that caused the failure. Looking at the source (Figure 12), we see that the problem was that the QUIT intrinsic was called. Notice that the call to RESETCRITICAL is commented out. (We contrived the failure, remember?). Thus, the problem. The process was still SETCRITICAL when the process quit, hence the SF311.

For the user with an application problem that directly causes a failure, the above steps summarize the basic steps necessary to diagnose this failure from a dump. As long as

the application source is available, it should be relatively simple to find the exact line of code which is causing the problem.

Figures 13 and 6 show an example of the second type of problem, a system loop. This dump was generated by writing a simple program which entered the linear queue, then went into an infinite loop. The halt button was pressed. Notice on the register page that the Current Process pointer is non-zero, the current CST is %301 (program segment), and the current instruction (CIR register) is %140000, which is a BR P-0 instruction. (You can't get a loop much tighter than that!). Looking at the PCB (Figure 6), we see the current (starred) PCB is at priority %30, indicating a linear queue. From there, all that would be necessary would be to look at the stack markers to determine the exact nature of the loop.

The third example was generated by starting several programs doing disc IO's, then taking the disc offline. This simulated a hardware disc hang. The CIR on the register page (Figure 14) is %030020 which is the PAUS instruction. If this instruction is present, then that means the hardware was not being asked to perform any work. If it is known that the application was indeed trying to run, then the obvious problem was that it could not for some reason. Determining the reason is sometimes very difficult, but there are a few things that can be checked:

1. Check the PCB. Check what wait bits are set. If the SW bit is on for several processes (Short Wait) then they were trying to perform disc IO's, and the DRQ and disc subsystem should be checked.

2. Check the SIR tables for SIR deadlocks. If this is so, then the problem is most likely an involved MPE bug, unless PM code is involved.

3. If the PCB shows the processes to be waiting for Global Rins (RG bit on) then the problem is most likely a file locking deadlock. This occurs when multiple files are being locked. (MR capability granted).

If we examine the Disc Request Queue (Figure 15) we see that there are several processes waiting to perform disc IO's to LDEV 1. Looking at the Device Information Table for that LDEV (Figure 16), we can see that the device is offline. This is determined by examining the hardware status words and interpreting them from the CE handbook.

The port lockout is considered a subset of the hang. A separate example will not be given.

The fourth and last example shows what happens when Priv Mode Debug is used to modify absolute memory location 0 to the value 0. (This was used to generate this failure). This particular memory location is used by the microcode to delimit the beginning of the CST table. If this points to a place that does not look like the CST, the microcode will halt the system. The main thing to look for in this type of dump is the CIR value. The type of instruction being executed would indicate what exactly was wrong. This is done by understanding exactly what the particular instruction does, and knowing what data in memory is used by that instruction. This tells what the actual instruction was that halted the system. Here, in Figure 17, we see that the current instruction was %31051, which is a PCAL 51. The PCAL instruction must examine the CST in order to determine the location in memory to branch to. We can see on the register page that low memory location for the CST pointer was zero. If we were to further examine the TBUF's (not shown), we would see the MA0 command, plus the DEBUG PRIV... message. This would be a sure fire tipoff as to the exact source of the problem.

Summary

This paper has attempted to give a very simple description of a very complex area of the HP3000. It should be apparent that the most important attribute that one can have when attempting to read and interpret a memory dump is a great understanding of the workings of MPE. The greater the understanding, the easier the dump is to read. Beyond this, familiarity with the exact format and listings produced by DPAN4 is an invaluable aid. Finally, a great deal of experience must be acquired before one can truly be considered an effective dump reader and interpreter.

```
HP3000 III MEMORY DUMPC 0:) 05 OF SYS VER C  UPDATE  E0  FIX 20  DUMP TIME  2/09/83,  5:08AM                    PAGE  2
(C) HEWLETT-PACKARD CO. 1980

                                        ******    REGISTERS    ******

.....................................................................................................................
:  DATA SEGMENT     :  CODE SEGMENT    :  MISCELLANEOUS :  STATUS : 103033  :  ISR : 100000                SERIES 30/33 :
:                   :                  :                :                  :                                            :
:  DB BANK : 000004 :  PB    : 073020 :  X    : 000000 :  MODE       : PRIV :  RUN/HALT  : RUN  STACK OVR  : OFF      :
:  DB      : 010623 :  P     : 103674 :  CIR  : 030377 :  INTERRUPTS : OFF  :  IRQ       : ON   BNDS OVR/UNF : OFF      :
:  S BANK  : 000004 :  PL    : 115347 :  NIR  : 000000 :  TRAPS      : OFF  :  CSRQ      : OFF  VIOL CODE   : NONE     :
:  DL      : 010467 :  PBBANK: 000000 :                 :  STACK OP   : LEFT :  PARITY    : OFF  DISABLE ATN : OFF      :
:  Q       : 016525 :  (P-PB): 010654 :                 :  OVERFLOW   : OFF  :  POWERFAIL : OFF                       :
:  S       : 016534 :                 :                 :  CARRY      : ON   :  POWERON   : OFF                       :
:  Z       : 021206 :                 :                 :  COND CODE  : CCE  :  DISP FLAG  : OFF                       :
:                   :                  :                :  SEGMENT #  :  33  :  ICS FLAG   : OFF                       :
.....................................................................................................................

********* HALT  17


                                        ******    FIXED LOW MEMORY    ******

(ADDR   0)   CODE SEGMENT TABLE POINTER                  030110
(ADDR   1)   EXTENDED CODE SEGMENT TABLE POINTER         100000
(ADDR   2)   DATA SEGMENT TABLE POINTER                  024110
(ADDR   3)   PROCESS CONTROL BLOCK BASE                  037510
(ADDR   4)   CURRENT PCB POINTER                         040210
(ADDR   5)   INTERRUPT STACK BASE                        041610
(ADDR   6)   INTERRUPT STACK LIMIT                       042606
(ADDR   7)   INTERRUPT MASK                              067600
(ADDR  10)   DRT BANK                                    000000
(ADDR  11)   DRT ADDR                                    000000
```

Figure 1

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
1	ININ	PRIV	ON	OFF	4674	184524	0				S	C
2	FILESYS1 (0)	PRIV	ON	OFF	10774	006623	7				S	
3	FILESYS4 (1)	PRIV	ON	OFF	3574	166623	1				S	
4	FILESYS5 (2)	PRIV	ON	OFF	4470	162023	1				S	
5	FILESYS6 (3)	PRIV	ON	OFF	5430	154223	1				S	
6	FILESYS6A (4)	PRIV	ON	OFF	12504	034023	7				S	
7	FILESYS7 (5)	PRIV	ON	OFF	6100	157223	7				S	
10	CIALTORG (6)	PRIV	ON	OFF	10570	104023	5				S	
11	CICOMSYS (7)	PRIV	OFF	OFF	4220		1	38573			S	
12	CIERR (10)	PRIV	ON	OFF	2700	113023	1				S	
13	CIFILEB (11)	PRIV	OFF	OFF	10750	141623	7				S	
14	CIFILEM (12)	PRIV	ON	OFF	3304	174223	1				S	
15	CIINIT (13)	PRIV	ON	OFF	7644	157623	5				S	
16	CILISTF (14)	PRIV	OFF	OFF	6500	017423	2				S	
17	CIMISC (15)	PRIV	OFF	OFF	4504		1	37064			S	
20	CIORGMAN (16)	PRIV	ON	OFF	6310	044423	3				S	
21	CIPREPRUN (17)	PRIV	ON	OFF	6424	166423	2				S	
22	CISUBS (20)	PRIV	ON	OFF	4520	142023	5				S	
23	CISYSMGR (21)	PRIV	OFF	OFF	7624		1	37240			S	
24	CIUSERUTIL (22)	PRIV	ON	OFF	4540	065023	5				S	
25	CXSTOREST (23)	PRIV	OFF	OFF	6440	062623	3				S	
26	RESTORE (24)	PRIV	OFF	OFF	6640	025223	1				S	
27	STORE (25)	PRIV	OFF	OFF	11744	000023	5				S	
30	DIRC (26)	PRIV	ON	OFF	7510	046623	7				S	
31	ALLOCATE (27)	PRIV	ON	OFF	6454	125023	5				S	
32	ALLOCUTIL (30)	PRIV	ON	OFF	6434	000023	7				S	
33	MAPDRES (31)	PRIV	ON	OFF	22330	073020	0				S	C
34	TERMRES (32)	PRIV	ON	OFF	20050	115350	0				S	C
35	ABORTDUMP (34)	PRIV	ON	OFF	7060	012223	5				S	
36	MESSAGE (35)	PRIV	ON	OFF	5000	021423	6				S	
37	PROCSEG (36)	PRIV	ON	OFF	7870	024023	7				S	
40	SOFTIO (37)	PRIV	OFF	OFF	20260		1	40267			S	
41	NRIO (40)	PRIV	ON	OFF	14664	137223	1				S	
42	PCREATE (41)	PRIV	ON	OFF	10140	152223	6				S	
43	MORGUE (42)	PRIV	ON	OFF	4600	185423	7				S	
44	BIPC (43)	PRIV	ON	OFF	3524	152623	7				S	
45	IPC (44)	PRIV	ON	OFF	11710	042423	5				S	
46	CHECKER (45)	PRIV	ON	OFF	1764	127823	1				S	
47	UTILITY1 (46)	PRIV	ON	OFF	5014	132023	1				S	
50	UTILITY2 (47)	PRIV	OFF	OFF	6664		1	40717			S	
51	RINS (51)	PRIV	ON	OFF	3644	020023	7				S	
52	JOBTABLE (52)	PRIV	ON	OFF	5150	105623	4				S	
53	DEBUG (53)	PRIV	ON	OFF	20554	021423	5				S	
54	NURSERY (54)	PRIV	OFF	OFF	7570		1	41171			S	
55	SPOOLING (57)	PRIV	OFF	OFF	23150	105423	6				S	
56	SPOOLCOMS1 (60)	PRIV	OFF	OFF	10350	163223	6				S	
57	SPOOLCOMS2 (61)	PRIV	OFF	OFF	12010		1	41513			S	
60	PVCOMSEG (62)	PRIV	OFF	OFF	3610		1	41566			S	

Figure 2

326	14	USER	OFF	OFF	5620		1	1123754		
327	14	USER	OFF	OFF	4100		1	1124004		
330	14	USER	OFF	OFF	5704		1	1124025		
331	14	USER	OFF	OFF	11610		1	1124055		
332	14	USER	OFF	OFF	13230		1	1124125		
333	14	USER	OFF	OFF	8670		1	1124203		
334	14	USER	OFF	OFF	12510		1	1124237		
335	14	USER	OFF	OFF	15760		1	1124312		
336	14	USER	OFF	OFF	4704		1	1124402		

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	R O C	I M I	S Y S	C R E S
301	15	USER	OFF	OFF	17734		1	1102157				
302	15	USER	OFF	OFF	17520		1	1102257				
303	15	USER	OFF	OFF	17714		1	1102356				
304	15	USER	OFF	OFF	17740		1	1102456				
305	15	USER	OFF	OFF	17700		1	1102556				
306	15	USER	OFF	OFF	5424		1	1102656				
307	15	USER	OFF	OFF	16544		1	1102705				
310	15	USER	OFF	OFF	17010		1	1103000				
311	15	USER	OFF	OFF	16644		1	1103075				
312	15	USER	OFF	OFF	16714		1	1103171				
313	15	USER	OFF	OFF	14744		1	1103265				
314	15	USER	OFF	OFF	17104		1	1103351				
315	15	USER	OFF	OFF	16030		1	1103446				
316	15	USER	OFF	OFF	15420		1	1103537				
317	15	USER	OFF	OFF	16514		1	1103626				
320	15	USER	OFF	OFF	17604		1	1103721				
321	15	USER	OFF	OFF	17424		1	1104021				
322	15	USER	OFF	OFF	16754		1	1104120				
323	15	USER	OFF	OFF	17424		1	1104214				
324	15	USER	OFF	OFF	17714		1	1104313				
325	15	USER	OFF	OFF	17550		1	1104413				
326	15	USER	OFF	OFF	17524		1	1104512				
327	15	USER	OFF	OFF	16434		1	1104611				
330	15	USER	OFF	OFF	12504		1	1104704				

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	R O C	I M I	S Y S	C R E S
301	16	USER	OFF	OFF	17314		2	454706				
302	16	USER	OFF	OFF	17624		2	455004				
303	16	USER	OFF	OFF	17410		2	455104				
304	16	USER	OFF	OFF	17534		2	455203				
305	16	USER	OFF	OFF	17510		2	455302				
306	16	USER	OFF	OFF	17564		2	455401				
307	16	USER	OFF	OFF	17474		2	455500				

Figure 3

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWP	SYS	CES	WD	VM ALLOC
1	(CODE SEGMENT TABLE)	OFF	1400	030110	0								S	C		0
2	(DATA SEGMENT TABLE)	OFF	4000	024110	0								S	C		0
3	(PROCESS CONTROL BLOCK)*	OFF	2000	037510	0								S	C		0
4	(CST EXTENSION)	OFF	6000	031510	0								S	C		0
5	(SYSTEM GLOBAL AREA)	OFF	200	001000	0								S	C		0
6	(FIXED LOW CORE)	ON	4000	000000	0								S	C		0
7	(INTERRUPT CONTROL STACK)	OFF	1100	041510	0								S	C		0
10	(SYSTEM BUFFERS)	ON	2020	057514	0								S	C		0
11	(UCOP REQUEST QUEUE)	ON	64	177623	7	>							S			1
12	(PROCESS-PROCESS COMMUNICATION TABLE)	ON	200	141423	5								S			1
13	(I/O QUEUE)	OFF	1310	042610	0								S	C		1
14	(TERMINAL BUFFERS)	OFF	17750	002130	0								S	C		0
15	(LOGICAL-PHYSICAL DEVICE TABLE)	ON	154	070210	0								S	C		0
16	(LOGICAL DEVICE AND CLASS TABLE)	ON	1234	172623	1								S	C		2
17	(DRIVER LINKAGE TABLE)	OFF	230	000440	0								S	C		0
20	(I/O RESOURCE TABLES)	OFF	14	000670	0								S	C		0
21	(SECONDARY MSG TABLE)	OFF	200	065320	0								S	C		0
22	(LOADER SEGMENT TABLE)	ON	3744	103223	2								S			30
23	(TIMER REQUEST LIST)	OFF	144	070364	0								S	C		0
24	(DIRECTORY)	ON	2000	103223	6								S			3
25	(DIRECTORY SPACE)	ON	600	177023	5								S			1
26	(RIN TABLE)	ON	504	030623	2											1
27	(SWAPTABLE)	OFF	2400	061534	0								S	C		0
30	(JOB PROCESS COUNT)	ON	20	000750	0								S	C		0
31	(JOB MASTER TABLE)	ON	200	182423	6								S			14
32	(TAPE LABEL TABLE)	ON	710	176223	4								S			1
33	(LOG TABLE)	OFF	650		1	6136	D						S			1
34	(REPLY INFORMATION TABLE)	OFF	2000		1	6237	D						S			3
35	(VOLUME TABLE)	ON	144	177023	0								S			1
36	(BREAKPOINT TABLE)	OFF	550		1	7117	D						S			1
37	(LOG BUFFER 1)	OFF	400	173023	3								S			1
40	(LOG BUFFER 2)	ON	400	057023	4								S			1
41	(LOG ID TABLE)	OFF	560		1	6132	D						S			1
42	(ASSOCIATION TABLE)	OFF	564	173623	6								S			1
43	(CST BLOCK)	OFF	120	064134	0								S	C		0
44	(JOB CUTOFF TABLE)	OFF	74	070530	0								S	C		0
45	(SYSTEM JIT)	ON	100	012023	5								S			1
46	(SPECIAL REQUEST TABLE)	OFF	144	064254	0								S	C		0
47	(VIRTUAL DISK SPACE TABLE)	OFF	344	065640	0								S	C		0
51	(ARSBM TABLE)	OFF	44	000704	0								S	C		0
52	(ILT)	OFF	11354	046140	0								S	C		0
53	(SIR TABLE)	OFF	170	070624	0								S	C		0
54	(FILE MULTI-ACCESS VECTOR)	ON	200	177223	4								S			2
55	(INPUT DEVICE DIRECTORY)	ON	400	174623	6								S			40
56	(OUTPUT DEVICE DIRECTORY)	ON	400	133623	5								S			40
57	(WELCOME MESSAGE #1)	OFF	1750		1	6767	D						S			2

Figure 4

HP3000 III MEMORY DUMPC.00.05 OF SYS VER C UPDATE E0 FIX 20 DUMP TIME 2/06/83. 2:28AM PAGE 3
(C) HEWLETT-PACKARD CO. 1980

****** PROCESS CONTROL BLOCK (1ST HALF) ******

WAIT STATE

	DATA SEGMENTS				FAMILY TREE				WAKEMASK																EVENTFLAGS																PSEUDO INTERRUPTS									MISC		
PIN	XDS	ADB	STK	OVALC	FTHR PIN	SON PIN	BRO PIN	OA	M	RG	RL	MA	BIO	IO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM	M	RG	RL	MA	BIO	IO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM	PSIM	MK	SK	ST	HB	CY	BK	RITBK	PIOVR	PTYPE	CRIT	HSIR
1			105			2		S								J																									NORM									SYST		
2		A	74		1		3																									J									NORM									SYST		
3			75		1		4							I			T	M																							NORM									SYST		
4			76		1		5	F								J																									NORM									SYST		
5			77		1		6	F													I																				NORM									SYST	C	
6			100		1		7	F								J																									NORM									SYST		
7	11		101		1	37	10	F								J																									NORM									SYSTU		
10			102		1		11									J								M																	NORM									SYST		
11			103		1		12							I		J																									NORM									SYST		
12			104		1		14									J																									NORM									SYST		
14			120		1		15	S											S	F																					NORM									SYST		
15			117		1			S											S	F																					NORM									SYST		
37			121		J	64		S											S																						NORM									UMAIN		
64 •			133		37			F																																	NORM									USONM		

Figure 5

HP3000 III MEMORY DUMPC.00.05 OF SYS VER C UPDATE E0 FIX 20 DUMP TIME 2/06/83 2:28AM PAGE 4
(C) HEWLETT-PACKARD CO. 1980

****** PROCESS CONTROL BLOCK (2ND HALF) ******

	SCHEDULING INFORMATION																					RESOURCES					LIFE/DEATH			MISCELLANEOUS						
PIN	NOPIN	POPIN	DISPQ	LQ	CQ	DQ	EQ	INTER	CORER	PRI	HIPRI	USEDQ	TRW	SW	LW	MP	PC	IPEXP	MSPRI	SAR	SOV	CRIT	HSIR	PREV IMPD PIN	NEXT IMPD PIN	SC	LIVE	DEAD	FAC	BMS	PPC	PCST	PBXPTR	SLLPTR	BPT LNK	SYSTEM PROC NAME
1				L						61																	L			SNF	NUL		10	60673		PROGEN
2		64	D	L						62																	L			SNF	NUL			60541		SYSIO
3				L						170																	L			SNF	NUL			60553		IOMESS
4				L						62																	L			SNF	NUL		1	60565		LOG
5				L						175												C					L			SNF	NUL		2	60577		MEMLOG
6				L						175																	L			SNF	NUL		3	60611		
7				L						175																	L			SNF	NUL		4	60623		UCOP
10				L						12										S							L			SNF	NUL		5	60635		PFAIL
11				L						175																	L			SNF	NUL		6	60647		DEVREC
12				L						216																	L			SNF	NUL		7	60661		LOAD
14				L						230					L												L			SNF	NUL			61513		
15				L						230					L												L			SNF	NUL			61537		
37					C			I		230					L											S	L			SNF	NUL			62162		
64 •	2		D	L				I		30																	L		F	SNF	NUL		28	60712		

100 ENTRYS
61 UNASSIGNED ENTRYS
17 ASSIGNED ENTRYS

Figure 6

HP3000 III MEMORY DUMPC 00.05 OF SYS VER C UPDATE EG FIX 20 DUMP TIME 2/09/83, 8:32AM PAGE 26
(C) HEWLETT-PACKARD CO 1980

TABLE INDEX	LOGICAL DEVICE	PCB	ADDR REL	DST	BUFFER ADDRESS	FUNC	COUNT	PARM1	PARM2	MISC	FLAGS.....	STATUS DESCRIPTION	STATUS
357	20	23	+08	134	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
344	20	30	+08	141	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
331	20	30	+08	141	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
316	20	26	+08	137	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
303	20	26	+08	137	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
270	20	30	+08	141	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
10	20	30	+08	141	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
242	20	23	+08	134	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
227	20	23	+08	134	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
214	20	30	+08	141	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
153	20	30	+08	141	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
140	22	34	+08	152	0	000011	0W	000000	000000	000000	007000 IW BL CO	NORMAL COMPLETION	1
77	22	34	$BUF	10	0	000005	0W	000017	000000	000000	017000 SB IW BL	NORMAL COMPLETION	1
125	22	34	+08	152	0	000021	0W	000000	000000	000000	007000 IW BL CO	NORMAL COMPLETION	1
112	22	34	+08	152	0	000015	0W	000000	000000	000000	007000 IW BL CO	NORMAL COMPLETION	1
64	22	34	+08	152	337	WRITE	13B	000320	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
23	22	34	+08	152	410	WRITE	116B	000320	000004	000032	007000 IW BL CO	NORMAL COMPLETION	1
51	20	30	+08	141	445	READ	0W	000001	000000	000043	007000 IW BL CO	NORMAL COMPLETION	1
36	20	30	+08	141	445	WRITE	1W	000053	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
1262	22	34	+08	152	337	WRITE	122B	000320	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1
1275	22	34	+08	152	532	WRITE	23B	000320	000004	000000	007000 IW BL CO	NORMAL COMPLETION	1

****** I/O REQUEST TABLE (IN USE LIST) ******

TABLE INDEX	LOGICAL DEVICE	PCB	ADDR REL	DST	BUFFER ADDRESS	FUNC	COUNT	PARM1	PARM2	MISC	FLAGS.....	STATUS DESCRIPTION	STATUS
42755	22	34	+08	152	337	READ	1B	100001	000000	000002	006000 IW BL	PENDING	0

Figure 7

******** DISC REQUEST TABLE ******** (AVAILABLE LIST)

STATUS: 0.XX -> PENDING
1.XX -> SUCCESSFUL
2.XX -> END OF FILE
3.XX -> UNUSUAL CONDITION
4.XX -> IRRECOVERABLE ERROR

TABLE INDEX	LDEV	UNIT	PCB	S S	DST/ BANK	OFFSET/ ADDRESS	FUNC	XFER CNT	PARM1	PARM2	MISC	SEG IDENT	SEGDSP	NXTAVL	- F L A G S - MAIN	AUX	STATUS
002000	2	0	26	S	137	000023	WRITE	1	000000	060647	000001			0	007010	003310	1 0
000120	2	0	23	S	134	000023	WRITE	1	000000	060617	000001			2000	007010	001430	1 0
001560	2	0	26	S	137	000023	WRITE	1	000000	060646	000001			120	007010	003070	1 0
000260	2	0	23	S	134	000023	WRITE	1	000000	060616	000001			1560	007010	001570	1 0
000040	2	0	26	S	137	000023	WRITE	1	000000	060645	000001			260	007010	001350	1 0
000300	2	0	23	S	134	000023	WRITE	1	000000	060615	000001			40	007010	001610	1 0
001180	2	0	26	S	137	000023	WRITE	1	000000	060644	000001			300	007010	002470	1 0
000360	2	0	23	S	134	000023	WRITE	1	000000	060614	000001			1180	007010	001670	1 0
000700	2	0	26	S	137	000023	WRITE	1	000000	060643	000001			360	007010	002210	1 0
001540	2	0	23	S	134	000023	WRITE	1	000000	060613	000001			700	007010	003050	1 0
001700	2	0	26	S	137	000023	WRITE	1	000000	060642	000001			1540	007010	003210	1 0
001140	2	0	23	S	134	000023	WRITE	1	000000	060623	000001			1700	007010	002450	1 0
001720	2	0	26	S	137	000023	WRITE	1	000000	060641	000001			1140	007010	003230	1 0
001520	2	0	23	S	134	000023	WRITE	1	000000	060622	000001			1720	007010	003030	1 0
001000	2	0	26	S	137	000023	WRITE	1	000000	060651	000001			1520	007010	002310	1 0
001600	2	0	23	S	134	000023	WRITE	1	000000	060621	000001			1000	007010	003110	1 0
001020	2	0	26	S	137	000023	WRITE	1	000000	060650	000001			1600	007010	002330	1 0
000720	2	0	23	S	134	000023	WRITE	1	000000	060620	000001			1020	007010	002230	1 0
000240	2	0	26	S	137	000023	WRITE	1	000000	060647	000001			720	007010	001550	1 0
001200	2	0	23	S	134	000023	WRITE	1	000000	060617	000001			240	007010	002510	1 0
000420	2	0	26	S	137	000023	WRITE	1	000000	060646	000001			1200	007010	001730	1 0
000520	2	0	23	S	134	000023	WRITE	1	000000	060616	000001			420	007010	002030	1 0
001420	2	0	26	S	137	000023	WRITE	1	000000	060645	000001			520	007010	002730	1 0
000460	2	0	23	S	134	000023	WRITE	1	000000	060615	000001			1420	007010	001770	1 0
001060	2	0	26	S	137	000023	WRITE	1	000000	060644	000001			460	007010	002370	1 0
000200	2	0	23	S	134	000023	WRITE	1	000000	060614	000001			1060	007010	001510	1 0
000740	2	0	26	S	137	000023	WRITE	1	000000	060643	000001			200	007010	002250	1 0
001760	2	0	23	S	134	000023	WRITE	1	000000	060613	000001			740	007010	003270	1 0
000760	2	0	26	S	137	000023	WRITE	1	000000	060642	000001			1760	007010	002270	1 0
001240	2	0	23	S	134	000023	WRITE	1	000000	060623	000001			760	007010	002550	1 0
000020	2	0	26	S	137	000023	WRITE	1	000000	060641	000001			1240	007010	001330	1 0
000500	2	0	23	S	134	000023	WRITE	1	000000	060622	000001			20	007010	002010	1 0
000440	2	0	26	S	137	000023	WRITE	1	000000	060651	000001			500	007010	001750	1 0
001340	2	0	23	S	134	000023	WRITE	1	000000	060621	000001			440	007010	002650	1 0
000140	2	0	26	S	137	000023	WRITE	1	000000	060650	000001			1340	007010	001450	1 0
000160	2	0	23	S	134	000023	WRITE	1	000000	060620	000001			140	007010	001470	1 0
000640	2	0	26	S	137	000023	WRITE	1	000000	060647	000001			160	007010	002150	1 0
001320	2	0	23	S	134	000023	WRITE	1	000000	060617	000001			640	007010	002630	1 0
000560	2	0	26	S	137	000023	WRITE	1	000000	060646	000001			1320	007010	002070	1 0
001440	2	0	23	S	134	000023	WRITE	1	000000	060616	000001			560	007010	002750	1 0
000600	2	0	26	S	137	000023	WRITE	1	000000	060645	000001			1440	007010	002110	1 0
001260	2	0	23	S	134	000023	WRITE	1	000000	060615	000001			600	007010	002570	1 0
001100	2	0	26	S	137	000023	WRITE	1	000000	060644	000001			1260	007010	002410	1 0

Figure 8

```
HP3000 III MEMORY DUMPC.00.05 OF SYS VER C  UPDATE  E0  FIX 20  DUMP TIME  2/08/83.  2:28AM              BANK   1        PAGE  170
(C) HEWLETT-PACKARD CO. 1980

                     ******    PCBX AND STACK MARKERS FOR DST 104  (PCB 12 )    ******

SEG REL   SEG REL    JMAT      JPCNT     JOB INPUT   JOB OUTPUT  JDT DST  JIT DST                                               JCUT
DL        DB         INDEX     INDEX     LOG DEV8    LOG DEV #   INDEX    INDEX     JOB TYPE   DUPLICAT  INTERACT  INIT Q     INDEX
000444    001444         0         0       20           20         63        45     UNDEF         YES       YES    001145        0

ADDRESS    BANK        X        DELTA P   STATUS    DELTA Q     SEGMENT                    OFFSET/PROCEDURE          MOD/PRODUCT

075300      1        177756    020214    102075    000012       75 KERNELC (100)
075266      1        037470    017521    101075    000014       75 KERNELC (100)
075252      1        000013    000767    141301    000012      301  USER SEGMENT
075240      1        000000    000000    140043    000004       43 MORGUE (42)

$$$$$$$$ DST  104 (STACK)                            $$$$$$$$

******PCBX. ****************************************************************************************************
***PXGLOBAL:
072423  000444 001444 177777 000024 000024 000063 008045 000000
***PXFIXED:
072433: 000120 001213 003474 001145 001000 177777 000000 000000  072443: 000000 000000 000000 000000 000000 040000 000000 000000
072453  000000 000000 000000 000000 043000 000000 000000 007674  072463: 000005 016302 000000 000000 000000 000000 000000 000000
072473  000000 000000 000000 000000 000000 000000 000000 000000  072503: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 072513 - 072532 SAME AS ABOVE

072533. 000000 000000 000000 000000 000000 000000 000000 000000  072543: 000000 000000 000000 000000 000000 000000 000000 000000
***PXFILE  (ZERO TABLE ENTRIES ARE NOT PRINTED)
072553  000310 000000 000000 000000 000000 000014 000000 000000  072563  000000 000000 000000 000000 000000 000000 000000 000000
072573  000246 000104 000100 000000 000000
------- FILE VECTOR TABLE:            ENTRY  ADDRESS  LOCK  BRK   LOCK COUNT/PIN   HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
072600. 000106 000000 000000 000000     0      106                    0   0
------- CONTROL BLOCKS:
072700(000105): 000001 100060 000001 051514 020040 020040 020040 000001 000766 000400 000200 000000   072700:...O..SL     .. ......
072714(000121): 000000 004000 000000 000000 000400 000000 000002 000000 000000 000000 015523 000000   072714:.................S..
072730(000135): 015523 177777 177777 000111 000000 000000 000001 000001 000000 000000 000000 000000   072730:.S....I.............
072744(000151)  000000 000001 000000 000001 000000 000000 000000 000000 000000 000000 000000 000000   072744:....................
072760(000165). 000000 000060 000003 043505 052120 051111 050040 002001 000767 000400 000200 000000   072760:...O.GETPRIP........
072774(000201): 000000 004000 000000 000000 000400 000000 000002 000000 000000 000000 000010 000000   072774:....................
073010(000215). 000010 177777 177777 006111 000000 000000 001401 000001 000000 000000 000000 000000   073010:.......I............
073024(000231)  000000 000002 000000 000002 000000 000000 000000 000000 000000 000000 000000 000000   073024:....................
073040(000245). 000000                                                                                073040:..
073041: 000000 000000 000000 000000 000000 000000
------- AVAILABLE FILE TABLE          FNUM    FTYPE    SNULL   PACB V      LACB V    IOQX
073053: 000000 000112 000000 000000     2      FILE             0  112      0    0
073057  000000 000104 000000 000000     1      FILE             0  104      0    0
**PXPOINTERS
073063  000000 000314 000434 000444
****DL REGISTER  ********************************************************************************************
073067(177000)  005507 042524 050122 044517 051111 008041 030001 001000 000000 044517 051111 052131   073067..GETPRIORI.!O...IORITY
073103(177014)  030001 001000 000000 005507 042524 050122 044517 051111 052131 030001 001000 000000   073103:O.....GETPRIORITYO....
073117(177030): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000   073117:....................
```

Figure 9

****** PRESENT STACKS ******

****** PCBX AND STACK MARKERS FOR DST 130 (PCB 24) ******
**** CURRENT PROCESS ****

SEG REL DL	SEG REL DB	JMAT INDEX	JPCNT INDEX	JOB INPUT LOG DEV#	JOB OUTPUT LOG DEV #	JDT DST INDEX	JIT DST INDEX	JOB TYPE	DUPLICAT	INTERACT	INIT Q.	JCUT INDEX
000444	000600	1	2	20	20	123	122	#S56	YES	YES	004575	0

ADDRESS	BANK	X	DELTA P	STATUS	DELTA Q	SEGMENT	OFFSET/PROCEDURE	MOD/PRODUCT
016525	4	000511	004706	140035	000144	35 ABORTDUMP (34)		
016361	4	000001	004170	140035	000007	35 ABORTDUMP (34)		
016352	4	000001	001510	140301	000207	301 USER SEGMENT		
016143	4	000000	001010	140430	000034	30 DIRC (26)		
016107	4	000002	001224	142430	000021	30 DIRC (26)		
016066	4	000004	001104	140430	000027	30 DIRC (26)		
016037	4	000064	000635	142430	000013	30 DIRC (26)		
016024	4	000064	004017	140301	000220	301 USER SEGMENT		
015604	4	000031	001010	140430	000034	30 DIRC (26)		
015550	4	000002	001224	142430	000021	30 DIRC (26)		
015527	4	000004	001104	140430	000027	30 DIRC (26)		
015500	4	000002	001224	142430	000021	30 DIRC (26)		
015457	4	000000	000614	140430	000013	30 DIRC (26)		
015444	4	000000	000247	140301	000020	301 USER SEGMENT		
015424	4	000000	000000	140043	000004	. 43 MORGUE (42)		

Figure 10

```
PROGRAM FILE BADLABEL PUB.GOERTZ

BADLABEL          0
   NAME          STT  CODE ENTRY SEG
   BADLABEL        1     0     0
   FOPEN          14                 ?
   XCONTRAP       15                 ?
   DIRECSCAN      16                 ?
   FCONTROL       17                 ?
   PRINTFILEINFO  20                 ?
   SORTINITIAL    21                 ?
   ASCII          22                 ?
   SORTOUTPUT     23                 ?
   QUIT           24                 ?
   SORTENO        25                 ?
   PRINT          26                 ?
   FCLOSE         27                 ?
   TERMINATE      30                 ?
   NOTAPE          2     0     3
   PROCESS'ENTRY   3  1024  1126
   EXCHANGEDB     31                 ?
   LUN            32                 ?
   ATTACHIO       33                 ?
   RELSIR         34                 ?
   RESETCRITICAL  35                 ?
   DASCII         36                 ?
   FWRITE         37                 ?
   SETCRITICAL    40                 ?
   GET'VOLUME'TABL  4  3445  3445
   GETSIR         41                 ?
   GET'GROUP       5  3634  3634
   MOUNT          42                 ?
   DISMOUNT       43                 ?
   CHECK'FSPACE'TA  6  4203  4203
   CHECK'DFSM      7  4337  4337
   LOAD'DSEGS     10  4741  4751
   GETDSEG        44                 ?
   LOADPROC       45                 ?
   GETUSERMODE    46                 ?
   LOAD'DSEGS'AGAI 11 4741  4757
   CONTROL'Y      12  6030  6030
   RESETCONTROL   47                 ?
   MINSTACK       13  6043  6043
   SEGMENT LENGTH     6120

PRIMARY DB        130    INITIAL STACK    1440    CAPABILITY        700
SECONDARY DB     4443    INITIAL DL          0    TOTAL CODE       6120
TOTAL DB         4573    MAXIMUM DATA    11810    TOTAL RECORDS      60
ELAPSED TIME  00:00:04.944                      PROCESSOR TIME  00:02.503
```

Figure 11

```
PAGE 0026   BADLABEL        BADLABEL <2.0>      Procedure PROCESS'ENTRY

01076000  00254 4                        * LAST FILE IS   *).2;
01077000  00266 4           ASSEMBLE(DUP,NOP);
01078000  00267 4           MOVE X := * FILE'NAME.(28);
01079000  00272 4           SCAN X UNTIL " " 1;
01080000  00274 4           LEN := TOS - @BBUFF;
01081000  00276 4           PRINT(BUFF,-LEN,X40);
01082000  00303 4           ERRORS'FOUND := TRUE;
01083000  00305 4       END;
01084000  00305 3       MOVE FILE'NAME := " ",2; MOVE X := FILE'NAME.(29);
01085000  00320 3       MOVE FILE'NAME := @ENT.(8);
01086000  00324 3       SCAN FILE'NAME UNTIL "." 1;
01087000  00327 3       MOVE X := "-",2;
01088000  00335 3       ASSEMBLE(DUP,NOP);
01089000  00336 3       MOVE X := GROUP'NAME.(8);
01090000  00341 3       SCAN X UNTIL "." 1;
01091000  00343 3       MOVE X := "-",2;
01092000  00351 3       ASSEMBLE(DUP,NOP);
01093000  00352 3       MOVE X := ACCOUNT'NAME.(8);

01094000  00355 3
01095000  00355 3       MOVE ADISC'ADDR := ENT(4).(2);
01096000  00362 3       VTAB'INDX := DA1.(0:8);
01097000  00365 3       IF  VTAB'INDX > 255 THEN LDEV2 := 0
01098000  00370 3           ELSE LDEV2 := LUN(VTAB'INDX.MVTABX);
01099000  00400 3       IF  <<NOT>> CHECK'LDEV(LDEV2) THEN BEGIN
01100000  00405 4           TOS := SIR;
01101000  00406 4           RELSIR(X.X);
01102000  00407 4        << RESETCRITICAL(0); >>
01103000  00407 4           CLEARBUFF;
01104000  00422 4           MOVE BBUFF := "INVALID VTABINDX FOR: ",2;
01105000  00443 4           ASSEMBLE(DUP,NOP);
01106000  00444 4           MOVE X := FILE'NAME.(28);
01107000  00447 4           SCAN X UNTIL " " 1;
01108000  00451 4           LEN := TOS - @BBUFF;
01109000  00453 4           PRINT(BUFF,-LEN,X40);
01110000  00460 4           ERRORS'FOUND := TRUE;
01111000  00462 4       <<  GO TO GET'LOST; >>  QUIT(1012);
01112000  00463 4       END;
01113000  00464 3
01114000  00464 3       LABEL'LDEV := LDEV2;
01115000  00466 3       ATTRET := ATTACHIO(LDEV2,0,0,@FLABEL,0,128,DA1.(8:8),DA2.1);
01116000  00502 3       IF  ATT1.(8:8) = 1 THEN GO TO GOOD'ATTACHIO;
01117000  00506 3
01118000  00506 3       IF  ATT1.(8:8) = X64 THEN BEGIN
01119000  00512 4           CLEARBUFF;
01120000  00527 4           MOVE BBUFF := "INVALID DIRECTORY ADDRESS FOR: ",2;
01121000  00555 4           ASSEMBLE(DUP,NOP);
01122000  00556 4           MOVE X := FILE'NAME.(28);
01123000  00561 4           SCAN X UNTIL " " 1;
01124000  00563 4           LEN := TOS - @BBUFF;
01125000  00565 4           PRINT(BUFF,-LEN,X40);
01126000  00572 4           TOS := SIR;
01127000  00573 4           RELSIR(X.X);
01128000  00574 4           RESETCRITICAL(0);
01129000  00576 4           ERRORS'FOUND := TRUE;
01130000  00600 4           GO TO GET'LOST;
01131000  00601 4       END;
01132000  00601 3
```

Figure 12

****** REGISTERS ******

DATA SEGMENT		CODE SEGMENT		MISCELLANEOUS		STATUS = 101301		ISR = 000300			SERIES 30/33
DB BANK	= 000005	PB	= 114623	X	= 000000	MODE	= PRIV	RUN/HALT	= HALT	STACK OVR	= OFF
DB	= 173023	P	= 114640	CIR	= 140000	INTERRUPTS	= ON	IRQ	= OFF	BNDS OVR/UNF	= OFF
S BANK	= 000005	PL	= 114652	NIR	= 000000	TRAPS	= ON	CSRQ	= OFF	VIOL CODE	= NONE
DL	= 172667	PBBANK	= 000007			STACK OP	= LEFT	PARITY	= OFF	DISABLE ATN	= ON
Q	= 173031	(P-PB)	= 000015			OVERFLOW	= OFF	POWERFAIL	= OFF		
S	= 173031					CARRY	= OFF	POWERON	= OFF		
Z	= 176263					COND CODE	= CCE	DISP FLAG	= OFF		
						SEGMENT #	= 101	ICS FLAG	= OFF		

****** FIXED LOW MEMORY ******

(ADDR 0)	CODE SEGMENT TABLE POINTER	030110
(ADDR 1)	EXTENDED CODE SEGMENT TABLE POINTER	000300
(ADDR 2)	DATA SEGMENT TABLE POINTER	024110
(ADDR 3)	PROCESS CONTROL BLOCK BASE	037510
(ADDR 4)	CURRENT PCB POINTER	041210
(ADDR 5)	INTERRUPT STACK BASE	041610
(ADDR 6)	INTERRUPT STACK LIMIT	042606
(ADDR 7)	INTERRUPT MASK	067600
(ADDR 10)	DRT BANK	000000
(ADDR 11)	DRT ADDR	000000

Figure 13

```
HP3000 III MEMORY DUMPC 00.05 OF SYS VER C  UPDATE  E0  FIX 20  DUMP TIME  2/10/83,  3:38PM                    PAGE  2
(C) HEWLETT-PACKARD CO. 1980

                                        ******   REGISTERS   ******

 DATA SEGMENT     CODE SEGMENT     MISCELLANEOUS   STATUS = 141075   ISR = 000303                          SERIES 30/33

 DB BANK = 000000  PB     = 135114  X    = 177756  MODE       = PRIV  RUN/HALT  = HALT  STACK OVR    = OFF
 DB      = 001000  P      = 140021  CIR  = 030020  INTERRUPTS = ON    IRQ       = OFF   BNDS OVR/UNF = OFF
 S BANK  = 000000  PL     = 163733  NIR  = 000000  TRAPS      = OFF   CSRQ      = OFF   VIOL CODE    = NONE
 DL      = 177777  PBBANK= 000000                  STACK OP   = LEFT  PARITY    = OFF   DISABLE ATN  = ON
 Q       = 041554  (P-PB)= 002705                  OVERFLOW   = OFF   POWERFAIL = OFF
 S       = 041636                                  CARRY      = OFF   POWERON   = OFF
 Z       = 042552                                  COND CODE  = CCE   DISP FLAG = ON
                                                   SEGMENT #  = 75    ICS FLAG  = ON

PAUSE INSTRUCTION IN CIR

                                        ******   FIXED LOW MEMORY   ******

(ADDR %0)   CODE SEGMENT TABLE POINTER                 030054
(ADDR %1)   EXTENDED CODE SEGMENT TABLE POINTER        000303
(ADDR %2)   DATA SEGMENT TABLE POINTER                 024054
(ADDR %3)   PROCESS CONTROL BLOCK BASE                 037454
(ADDR %4)   CURRENT PCB POINTER                        000000
(ADDR %5)   INTERRUPT STACK BASE                       041554
(ADDR %6)   INTERRUPT STACK LIMIT                      042552
(ADDR %7)   INTERRUPT MASK                             063610
(ADDR %10)  DRT BANK                                   000000
(ADDR %11)  DRT ADDR                                   000000
```

Figure 14

******** DISC REQUEST TABLE ******** (SUMMARY INFO)

TOTAL ENTRIES IN TABLE:	100
ENTRY SIZE:	20
ENTRIES IN PRIMARY AREA:	61
IMPEDED PROCESS PCB:	
TABLE INDEX OF FIRST AVAIL ENTRY:	620
TABLE INDEX OF LAST AVAIL ENTRY:	2000
MAXIMUM NUMBER OF ENTRIES IN USE:	13
CURRENT NUMBER OF ENTRIES IN USE:	13
OVERFLOWS:	
TOTAL REQUESTS:	13354
SYSBASE INDEX OF DISABLED Q HEAD:	
SYSBASE INDEX OF DISABLED Q TAIL:	

******** DISC REQUEST TABLE ******** (ACTIVE LISTS)

LDEV 1

STATUS: 0.XX -> PENDING
1.XX -> SUCCESSFUL
2.XX -> END OF FILE
3.XX -> UNUSUAL CONDITION
4.XX -> IRRECOVERABLE ERROR

TABLE INDEX	LDEV	UNIT	PCB	S S	DST/ BANK	OFFSET/ ADDRESS	FUNC	XFER CNT	PARM1	PARM2	MISC	SEG IDENT	SEGDSP	URGCLS	- F L A G S - MAIN	AUX	STATUS
001120*	1	0	25	S	138	000023	WRITE	1	000000	107240	000000			240	006110	002430	0 1
001460	1	0	3		116	000200	READ	1200	000000	062223	000000			170	006100	002770	0 0
000400	1	0	27	S	140	000023	WRITE	1	000001	004405	000000			240	006100	001710	0 0
001500	1	0	32	S	143	000023	WRITE	1	000001	007633	000000			240	006100	003010	0 0
001040	1	0	22	S	133	000023	WRITE	1	000000	107106	000000			240	006100	002350	0 0
001220	1	0	30	S	141	000023	WRITE	1	000001	007617	000000			240	006100	002530	0 0
000220	1	0	24	S	135	000022	READ	1	000000	107124	000000			242	006100	001530	0 0
001620	1	0	21	S	132	000023	WRITE	1	000000	107070	000000			242	006100	003130	0 0

Figure 15

```
          (021757)   0   022757     000000 000000 000000 000000 000000 000120 000000 000000
          (021767)   0   022767     000000 000000 000000 000000 000000

DRT NO  41                                                    CONTROLLER ERROR STATUS = 000000

          (MAGNETIC TAPE UNIT) -          TYPE 24     SUBTYPE 0

          UNIT 0  LOGICAL DEV 7    FLAGS = 002000    NEXT DIT = 000000  DLTP = 177450  ILTP = 054547  IOQP = 000000

          DIT
            (021774)   0   022774     002000 000000 000000 040007 177450 054547 000000 000000
            (022004)   0   023004     000000 000000 000000 000000

          (MAGNETIC TAPE UNIT)            TYPE 24     SUBTYPE 0

          UNIT 1  LOGICAL DEV 8    FLAGS = 002000    NEXT DIT = 000000  DLTP = 177450  ILTP = 054547  IOQP = 000000

          DIT
            (022010)   0   023010     002000 000000 000000 040410 177450 054547 000000 000000
            (022020)   0   023020     000000 000000 000000 000000

          (MAGNETIC TAPE UNIT)            TYPE 24     SUBTYPE 0

          UNIT 2  LOGICAL DEV 9    FLAGS = 002000    NEXT DIT = 000000  DLTP = 177450  ILTP = 054547  IOQP = 000000

          DIT
            (022024)   0   023024     002000 000000 000000 041011 177450 054547 000000 000000
            (022034)   0   023034     000000 000000 000000 000000

          (MAGNETIC TAPE UNIT)            TYPE 24     SUBTYPE 0

          UNIT 3  LOGICAL DEV 10   FLAGS = 002000    NEXT DIT = 000000  DLTP = 177450  ILTP = 054547  IOQP = 000000

          DIT
            (022040)   0   023040     002000 000000 000000 041412 177450 054547 000000 000000
            (022050)   0   023050     000000 000000 000000 000000

DRT NO  49

          (SYSTEM DISC)                   TYPE 0      SUBTYPE 8

          UNIT 0  LOGICAL DEV 1    FLAGS = 040010    NEXT DIT = 000000  DLTP = 177460  ILTP = 054745  IOQP = 044240

          DIT
            (022054)   0   023054     040010 000000 044240 040001 177460 054745 000000 000000
            (022064)   0   023064     044600 044740 000020 103240 000077 001440 034446 000001
            (022074)   0   023074     000001 000000 011400 103043 000077 001437 000000 000000
            (022104)   0   023104     000000 000000 000000 000000 000000 000000 000000 000000
            (022114)   0   023114     000001 000000 103043 000000 000000
```

Figure 16

****** REGISTERS ******

DATA SEGMENT		CODE SEGMENT		MISCELLANEOUS		STATUS = 142453		ISR = 000000				SERIES 30/33
DB BANK	= 000000	PB	= 017423	X	= 000003	MODE	= PRIV	RUN/HALT	= RUN	STACK OVR	= OFF	
DB	= 000000	P	= 037643	CIR	= 031051	INTERRUPTS	= ON	IRQ	= OFF	BNDS OVR/UNF	= OFF	
S BANK	= 000004	PL	= 040176	NIR	= 000005	TRAPS	= OFF	CSRQ	= OFF	VIOL CODE	= NONE	
DL	= 010467	PBBANK	= 000005			STACK OP	= LEFT	PARITY	= OFF	DISABLE ATN	= OFF	
Q	= 012014	(P-PB)	= 020220			OVERFLOW	= OFF	POWERFAIL	= OFF			
S	= 012014					CARRY	= ON	POWERON	= OFF			
Z	= 021473					COND CODE	= CCL	DISP FLAG	= OFF			
						SEGMENT #	= 53	ICS FLAG	= OFF			

****** FIXED LOW MEMORY ******

(ADDR %0)	CODE SEGMENT TABLE POINTER	000000
(ADDR %1)	EXTENDED CODE SEGMENT TABLE POINTER	000000
(ADDR %2)	DATA SEGMENT TABLE POINTER	024110
(ADDR %3)	PROCESS CONTROL BLOCK BASE	037510
(ADDR %4)	CURRENT PCB POINTER	040070
(ADDR %5)	INTERRUPT STACK BASE	041810
(ADDR %6)	INTERRUPT STACK LIMIT	042606
(ADDR %7)	INTERRUPT MASK	067600
(ADDR %10)	DRT BANK	000000
(ADDR %11)	DRT ADDR	000000

Figure 17

CONCEPTS OF TAPE MANAGEMENT IN THE HP3000 ENVIRONMENT

Joerg Groessler
Ingenieurburo Joerg Groessler

This presentation describes the history and concept of an automatic tape management system. The philosophy and principles can be understood best by following our long and tough way from a single-user and single-tape system to a big installation with severe problems in DP administration. Therefore I want to divide the story into six phases, each one representing one step towards a fully automated and quite comfortable tape management system.

PHASE 1: NO TAPE MANAGEMENT

This is the typical starting phase after installation of the new HP3000 system. The characteristics are:

- One account besides SYS and SUPPORT (names like 'GEORGE.TEST' are rather common).
- One tape, used every day for a full SYSDUMP (takes only 5 minutes...)
- Everything is in order (manuals - if existing - are stored properly, the only tape can be found on the desk of the DP manager - everybody knows that...)

duration of phase 1: normally until 3 days before the first HP training begins.

PHASE 2: A SIMPLE MANUAL TAPE MANAGEMENT

After the first user (do we really need them ??) complained that a very important file was lost because the STORE tape had been overwritten (some users have started procedures in addition to SYSDUMP, they call that 'backup') the DP manager decided that all tapes used should be written into a list.

This is the point where more than 20 tapes are used periodically and one person can hardly remember what is stored on all these tapes. A typical layout of a simple tape list can look like this (for better identification tapes get a unique number which is written on a label on the tape reel):

Tape #	date	descr
1	3/28/75	some files of PUB.ACCT1
3	3/29/75	SYSDUMP

As an additional feature, labels can be put on the tape containing more or less comments ('all files which I have modified today...'). The SYSDUMP now takes one hour and therefore is only performed once a week. After the first parity error during RELOAD (Mister Smith used an old version of a privileged program from the CL and crashed the system) an alternate tape set is used so that always two full SYSDUMPs are available.

duration of phase 2: some more weeks

PHASE 3: A MORE INTELLIGENT MANUAL TAPE MANAGEMENT

Once a user asked: "I lost a file, and after RESTORE I found a version within my group which must be at least two weeks old. Where can I find an actual version of this file ?".

It is obvious that the tape list of phase 2 contains not enaugh information to satisfy requests like this and others. The DP manager therefore decides to enhance his manual tape list (also the appearance became much better now):

tape #	function	user name	date	comment
2	STORE	TEST.ACCT1	3/10/76	Payroll source programs
10	SYSDUMP	PUB.SYS	3/11/76	Weekly SYSDUMP

PHASE 4: A COMPUTER SUPPORTED MANUAL TAPE MANAGEMENT

Phase 3 lasted for several years. But then the need for a computer supported tape management was discovered. The main reasons to write a program for tape insertion and deletion from a list were:

- There were more than 500 tapes handled (STORE, SYSDUMP, non-Store tapes). Of some tapes it was still hard to say what was on them (the operator sometimes forgot to write them into the list). Tape management took a considerable amount of time of the operator (30% and more).

- There was still not enough information in the list. The list itself was not enhanced to avoid an increase of the time the operator had to spend.

- There was still no information about <u>what files</u> were stored on STORE and SYSDUMP tapes.

- It was not understandable why the computer's administration should not be supported by the computer itself.

A program called REPPROG was implemented which created and maintained a tape list much more precise:

```
*** PAGE 3 ***  DATE: 25-03-83

TAPE  CREATION  EXPIRY   GROUP   ACCOUNT   FILENAMES
--------------------------------------------------------------------------------------------------------------
1     08-02-83           WEISS   ELEKTRON  ****
                         WEISS   ELEKTRON  ADU      BASICMOD BASICMOO BASICURO BASICURV BASIFLOO BASIFLOP BASURLAO BASURLHO
                                           BF08086  BF0URLA  BF0Z80   BF18080  BF180800 BF18086  BF180860 BFIFLOP  BFIFLOPO
                                           BF1URLA  BF1URLAO BF1Z80   BF1Z800  CHARDEF  CHARDEFO HEXCOD   HEXCODOB LASBIN
                                           LASKOR   LASMESS  LASOPT   LASOPT1  MABE     MABEOB   MAUR     MAUROB   MLA1
                                           MLA10B   MLA2     MLA20B   MLA2PR   MLA2PROB MLA3     MLA30B   MLA4     MLA40B
                                           MLA5     MLA50B   MLA6     MLA60B   MONI8080 MONI8086 MRB1     MRB10B   MRB2
                                           MRB20B   REGASM   RXV24P   SUPERB   TSIO     WER      WER2     WERLA    WERMOZ80
                                           ZEICHEN  ZEICHENO
2     15-04-82           KLAAS   ELEKTRON  ****
                         KLAAS   ELEKTRON  BETRAG1  BETRAG2  COMP     CONRIP   COOL     EINTON   FILTRI   FIR      FOURKO
                                           GENER    IMRES    ITER     KLIRR    KO       LETZTOPT LOG1     LOG2     MONO
                                           OPT12    OPT16    OPT64    OTTO     SI1      SI2      SIIN     SIMU     SININT
                                           SPEC     SPECTRUM SPRAUSCH TIME12   TIME64   TRANS    UEFUNK   VERSUCH  ZLETZT
3     15-04-82           BRAUNI  ELEKTRON  ****
                         BRAUNI  ELEKTRON  BCDDUAL  CAT      CATP     DREHREG  DREZAANZ DTRREG   DUALBCD  INITMSR  INITMSR
                                           LOADC    LOADER   LOADF    MONITOR  OR16781  OR8781   PUNCH    PUNCHER  WEISS
                                           XAECL    XAECLPH
4     02-06-81           INTEL   KURS      ****
                         INTEL   KURS      HP1
5     30-06-82           LOUIS   PRZ       ****
                         LOUIS   PRZ       BBALLSLA BMJO3    BMSPL    BMSPL1   BMTABGEN BMTABLE  BMUSER3  CHSUM    CHSUMS
                                           CONRTEN3 CONTMOD  INSTTBL  RTBCOM   RTBGJO   RTBSYS   RTEN3GJO SINUS    TMOD20LA
                                           TMODGJO
6     16-11-82           BADEN   WERKST    ****
                         BADEN   WERKST    BCFILE   BEINGQ   BLESEQ   BRECHQ   CONTMOD  DATSAM4  DATSAME  DATSAMS  DLESEQ
                                           DRECHEQ  DSCHLQ   DUMQ     DVR11S   EEING2Q  EEINGQ   EKORRP   EKORRQ   ELESEP
                                           ELESEQ   ERECH1Q  ERECH2Q  ERECHEQ  GAEHDQ   GBALKDQ  GBALKEQ  GESDAT4Q GESDATEQ
                                           GESDATSQ GESLES4Q GESLESEQ GESLESSQ GRECRQ   GURLINQ  M10A     M10AE    M10H
                                           M10HE    M10H     M10HE    M10W     M10WE    M15A     M15H     M15W     M35A
                                           M35AE    M35H     M35HE    M35H     M35HE    M35W     M35WE    M60A     M60H
                                           M60H     M60W     MACU     MACUE    MALH     MALHE    MALW     MALWE    MAMG
                                           MAMGE    MAZH     MAZHE    MC37H    MC37W    MC42     MC42E    MCUH     MCUHE
                                           MCUW     MCUWE    MECP     MECS     MEHP     MEHS     MMO      MP1      MP2
                                           MP3      MP4      MP5      MP6      MP7      MVIT     MX5H     MX5HE    MX5W
                                           MX5WE    PSTRANQ  RECUMQ   RTETMOD  S10A     S10AE    S10H     S10HE    S10H
                                           S10HE    S10W     S10WE    S15A     S15H     S15W     S35A     S35AE    S35H
                                           S35HE    S35H     S35HE    S35W     S35WE    S60A     S60H     S60H     S60W
                                           SACU     SACUE    SALH     SALHE    SALW     SALWE    SAMG     SAMGE    SAZH
                                           SAZHE    SC37H    SC37W    SC42     SC42E    SCUH     SCUHE    SCUW     SCUWE
                                           SECP2    SECS2    SEMP2    SEMS2    SMO      SMAPTMOD SP1      SP2      SP3
                                           SP4      SP5      SP6      SP7      SVIT     SX5H     SX5HE    SX5W     SX5WE
                                           TRANS    TRANSQ
```

And this was the method of REPPROG:

After a STORE or any other tape write operation had been performed, the tape was mounted again and REPPROG was started. The tape was entered using a 'NEW TAPE' command and then REPPROG read the directory of STORE and SYSDUMP tapes to get all filenames stored on the tape.

Very useful was an inquiry function, where the user could specify any 'wild card' filename and REPPROG listed all tapes and the date of their creation where this file could be found: @LPROT@.@.@

*** PAGE 1 *** DATE: 25-03-83

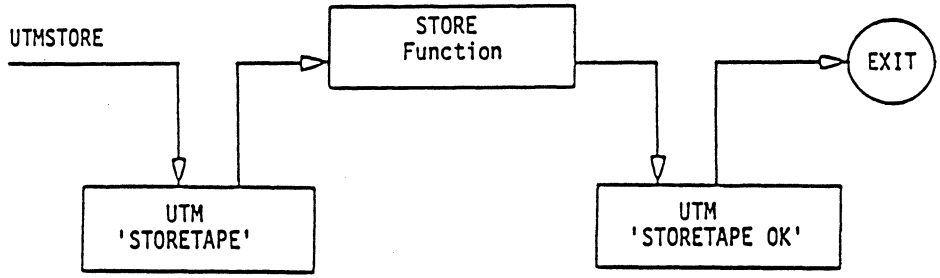
FILENAME	GROUP	ACCOUNT	ON TAPE	CREATED
LPROT	CLEMENS	PRZ	73	31-10-80
LPROT18S	SOURCE	PVSOFT	12	19-06-80
LPROT20S	SOURCE	PVSOFT	12	19-06-80
RTMLPROT	RTMVERB	SOURCES	64	23-02-81
YLPROT	PUB	WERKST	180	23-04-81
LPROT18S	SOURCE	PVSOFT	152	16-06-81
LPROT20S	SOURCE	PVSOFT	152	16-06-81
LPROT18R	WIESE	PRZ	45	14-08-81
LPROT18S	WIESE	PRZ	45	14-08-81
LPROTJ	WIESE	PRZ	45	14-08-81
LPROTENT	BERND	PRZ	38	14-08-81
LPROTSPC	BERND	PRZ	38	14-08-81
LPROT	REINER	PRZ	34	02-09-81
NLLPROT	REINER	PRZ	34	02-09-81
LPROT	REHBEIN	PEARL	523	09-09-81
NLLPROT	REINER	PRZ	58	25-01-82
CMDLPROT	BERLIN	PROJECT	803	25-11-75
COMLPROT	BERLIN	PROJECT	803	25-11-75
DLPROT	BERLIN	PROJECT	803	25-11-75
LPROTABS	BERLIN	PROJECT	803	25-11-75
LPROTGJO	BERLIN	PROJECT	803	25-11-75
LPROTJ	BERLIN	PROJECT	803	25-11-75
LPROTMS	BERLIN	PROJECT	803	25-11-75
LPROTS	BERLIN	PROJECT	803	25-11-75

The disadvantage of REPPROG still was that there was no possibility to enter tapes into the tape list automatically when they were used. So tapes were still not registed because someone forgot to run REPPROG after tape usage.

PHASE 5: UTM, THE FULLY AUTOMATIC TAPE MANAGEMENT

It took years until the problem of automatic tape management was technically solved.

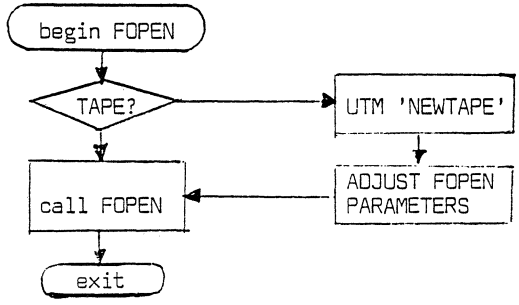
No problems with STORE and SYSDUMP: UTM can start the function by using the COMMAND intrinsic for STORE and RESTORE and CREATE for SYSDUMP:



No possibily however existed for managing tapes used by programs (like FCOPY).

Early 1982 I invented a method which is common on e.g. IBM systems but hard to implement within MPE: intrinsic trapping. It works as follows:

A user-written FOPEN and FCLOSE is called prior to the real intrinsic call. This special FOPEN now decides whether or not the FOPEN affects a tape and whether or not this tape has to be put into the tape list:



The handling of UTM is done by UDCs for STORE, RESTORE and SYSDUMP and by a slightly modified FILE command for the tape file equation:

```
:UTMSTORE "T34778;FN;NOTL;EXPDATE=3/28/83;@;@;@;*T;SHOW"
```

In the example above a UTM Store command is performed for tape number 34778, the filenames are put into the database, no tape labels are generated and no further write operation will be allowed for the tape until 28-3-83.

```
:FILE T=T34778;DEV=TAPE
:FCOPY FROM=*T;TO=MYFILE
```

This example shows how to access the same tape in an FCOPY operation.

The reports printed by UTM contain more interesting information than those created by REPPROG:

(1) - FR - free, STOL - STORE tape entered manually, STOK - UTMSTORE tape
(2) - VS - version of the specified qualifier, based on creation date
(3) - USAGE - incremented by one each time the tape is accessed

```
*** UTM  V.1.1 *** 03/25/83

 T.NO STAT QUALIFIER VS  USAGE  ORGDATE CREATDATE  EXPIRY    GROUP    ACCOUNT   FILENAMES
-------------------------------------------------------------------------------------------------------------
    1 STOK  HSYSDUMP  1       4  11/28/81 01/03/83 01/03/83 UTMDEMO  IJG        MONTHLY BACKUP (SYSDUMP), TAPE 1 (4 REELS)
                                                      PUB      IJG       ADRK     ADRLIST  ADRLISTS DSHDLS   EXPL      FILECOM
                                                                         FILECOMS REPPROG  REPPROGS
                                                      PUB      ACCT1     GENERIC  IDLEDOKU IDLEREPD IDLEREPS IDLES    TMSMON
                                                                         TMSMONA1
    2 STOK  HSYSDUMP  2       3  02/01/82 01/03/83 01/03/83. UTMDEMO  IJG        MONTHLY BACKUP (SYSDUMP), TAPE 2 (4 REELS)
                                                      PUB      IJG       ADRK     ADRLIST  ADRLISTS DSHDLS   EXPL      FILECOM
                                                                         FILECOMS REPPROG  REPPROGS
                                                      PUB      ACCT1     GENERIC  IDLEDOKU IDLEREPD IDLEREPS IDLES    TMSMON
                                                                         TMSMONA1
    3 STOK  WSYSDUMP  1       2  11/01/81 05/03/82 05/10/82 PUB      SYS        WEEKLY BACKUP (SYSDUMP), TAPE 1 (4 REELS)
                                                      GROUP1   ACCT1     REPPROG  REPPROGD REPPROGE TIERSPS  TMS053    TMS1
                                                                         TMSIN    TMS2     TMS264   TMS266    TMSJ      TMS4
                                                                         TMSCONF  TMSDOC   TMSDOCU  TMSJ
                                                      PUB      ACCT1     ADRK     BATCHJ1  BATCHJ2  CAT       DBINITJ   EMTPYCH
                                                                         F        HINWEISE INSTHINW HSYSDJ    AUTHCAT   PS
                                                                         SCHEMRES TCAT
                                                      PUB      ACCT2     CAT1     DATECONV DSDOCU   DSHDLS    DUMPUSL   FILECOM
                                                                         FILECOMS FILEERR  FORTCONV GENERIC   IDLEDOKU  IDLEREP
                                                                         IDLEREPS IDLES    TMSMON   TMSMONA1
                                                      PUB      IJG       ADRK     ADRLIST  ADRLISTS ALTACNT   ALTUSGR   BMMOLIS
                                                                         BREAKS   BREAKU   CAT      TMSMONA2  TMSMONS   TMSMONU
                                                                         TMSPROG  TMSPROGS TMSREP   TMSREPS   TMSSPEC   TMSUJOB
                                                                         TP       TRAFILE  TRAFILED TRAFILES  TS        UDC
                                                                         US   WFO  YPROG
                                                      UTM      IJG       UTMCAT   UTMCOMLJ UTMCOMPJ UTMDB1S   UTMDB2S   UTMDB2U
                                                                         UTMDBU   UTMENTRY UTMINST  UTMINSTS  UTMMSG1   UTMMSG2
                                                                         UTMSCHEM UTMSTOJ  UTMTRAEJ UTMTRAGJ  UTMTPAJ   UTMUTIL
                                                      UTMDEMO  IJG       UTMCAT   UTMDBS   UTMFORM  UTMMSG1   UTMMSG2   UTMSCHE
                                                                         UTMUTILS XXXX0000
                                                      UTMS     IJG       TRANSJ   TREC     URSCHEMA US        UTMCAT    UTMCOML
                                                                         UTMCOMPJ UTMDB1S  UTMDB1U  UTMDB2S  UTMDB2U   UTMDBCJ
                                                                         UTMDBJ   UTMDBU   UTMDBVS  UTMDBVU  UTMENTRY   UTMFORM
                                                                         UTMFORMN UTMFORMS UTMGR    UTMGR01  UTMGR02   UTMGR03
                                                                         UTMGR04  UTMGR05  UTMGR06  UTMGR07  UTMGR08   UTMGP09
                                                                         UTMGR10  UTMGR11  UTMINST  UTMINSTS UTMJ1     UTMMSG1
                                                                         UTMMSG2  UTMMS    UTMSCHEM UTMSF    UTMTRAEJ  UTMTRAG
                                                                         UTMTRAJ  UTMUTIL  UTMUTILS UTMVS   WSYSDJ
    4 FR    WSYSDUMP          1  02/01/82 04/19/82 04/26/82                    WEEKLY BACKUP (SYSDUMP), TAPE 2 (4 REELS)
    5 SYOL  WSYSDUMP  2       1  12/28/81 04/26/82 05/03/82                    WEEKLY BACKUP (SYSDUMP), TAPE 3 (4 REELS)
    6 STOK  UPDATE    1      15  01/02/82 12/31/82 12/31/82 UTMDEMO  IJG        DAILY UPDATE, TAPE 1
                                                      PUB      IJG       ADRK     ADRLIST  ADRLISTS DSHDLS   EXPL      FILECOM
                                                                         FILECOMS REPPROG  REPPROGS
                                                      PUB      ACCT1     GENERIC  IDLEDOKU IDLEREPD IDLEREPS IDLES    TMSMON
                                                                         TMSMONA1
    7 FR    UPDATE           12  01/02/82 11/18/82 11/18/82 UTMDEMO  IJG        blablabla
    8 FR    UPDATE           14  12/10/81 11/18/82 11/18/82 UTMDEMO  IJG        DAILY UPDATE, TAPE 3
    9 FR    UPDATE           13  12/10/81 11/18/82 11/18/82 UTMDEMO  IJG        DAILY BACKUP, TAPE 4
   10 FR    UPDATE           18  01/02/82 11/18/82 11/18/82 UTMDEMO  IJG        DAILY BACKUP, TAPE 5
   11 FR    UPDATE            7  02/01/82 11/18/82 11/18/82 UTMDEMO  IJG        DAILY BACKUP, TAPE 6
   12 FR    ACCT1             1  01/02/82 04/22/82 04/22/82                    ADDITIONAL BACKUP FOR ACCOUNT 1, TAPE 1
```

In addition some new features are included:

- the'generation method' allows the user to identify a tape by a name rather than a tape number when more than one tape set is used in a cyclic manner:

  ```
  :UTMSTORE "WEEKLY;FN;@;*T;SHOW
  ```

  a message appears on the operator's console:

  ```
  PLEASE MOUNT TAPE 'WEEKLY' NUMBER 45690
  ```

  During RESTORE a specific version can be selected:

  ```
  :UTMRESTORE WEEKLY,2;*T
  ```

- tape labels are generated containing the tape number as volume id. NOTL in UTMSTORE suppresses tape labels which is mandatory for the time being.

The filename inquiry was enhanced and the list now contains the volume number of the STORE tape where the file can be found:

```
*** UTH  V.1.1 *** 03/25/83

TAPENUMBER  VOLUME  CREATION   FILENAME    GROUP   ACCOUNT
-----------------------------------------------------------
         1       1  01/03/83   ADRLISTS    PUB     IJG
         1       1  01/03/83   ADRLIST     PUB     IJG
         1       1  01/03/83   ADRK        PUB     IJG
         3       1  05/03/82   ADRK        PUB     ACCT1
         3       1  05/03/82   ADRK        PUB     IJG
         3       1  05/03/82   ADRLIST     PUB     IJG
         3       1  05/03/82   ADRLISTS    PUB     IJG
        14       1  05/04/82   ADRK        PUB     ACCT1
       203       1  11/13/82   ADRLISTS    PUB     IJG
       203       1  11/13/82   ADRLIST     PUB     IJG
       203       1  11/13/82   ADRK        PUB     IJG
       101       1  11/12/82   ADRK        PUB     IJG
       101       1  11/12/82   ADRLIST     PUB     IJG
       101       1  11/12/82   ADRLISTS    PUB     IJG
         2       1  01/03/83   ADRK        PUB     IJG
         2       1  01/03/83   ADRLIST     PUB     IJG
         2       1  01/03/83   ADRLISTS    PUB     IJG
      5000       1  11/29/82   ADRK        PUB     IJG
      5000       1  11/29/82   ADRLIST     PUB     IJG
      5000       1  11/29/82   ADRLISTS    PUB     IJG
         6       1  12/31/82   ADRLISTS    PUB     IJG
         6       1  12/31/82   ADRLIST     PUB     IJG
         6       1  12/31/82   ADRK        PUB     IJG
       400       1  12/31/82   ADRLISTS    PUB     IJG
       400       1  12/31/82   ADRLIST     PUB     IJG
       400       1  12/31/82   ADRK        PUB     IJG
       301       1  11/29/82   ADRK        PUB     IJG
       301       1  11/29/82   ADRLIST     PUB     IJG
       301       1  11/29/82   ADRLISTS    PUB     IJG
        20       1  12/31/82   ADRLISTS    PUB     IJG
        20       1  12/31/82   ADRLIST     PUB     IJG
        20       1  12/31/82   ADRK        PUB     IJG
```

PHASE 6: UTM VERSION 2.0, WE WAIT FOR THE Q-MIT

Some enhancements and error corrections of the Q-MIT help UTM to do its job:

- Tape labels can be used with STORE and RESTORE (they will never work with SYSDUMP - why?)

- STORE and RESTORE are running as processes rather than proceduers within the command interpreter. This makes it possible to handle continuation reels as individual tapes. A problem still will be the labeled tape option because continuation reels are specially handled.

In addition, more information is stored in the database and new functions are added:

- The 'user security code' controls what tape operations and inquiry can be used by each user.

- UTMFORCE makes UTM mandatory for all tape operations (this will be an option which can be switched on and off by the operator).

- UTMTAPEUNAVAILABLE keeps track of tapes which are removed temporarily and should not be deleted from the list. An availibility code gives the user-defined reason why the tape is unavailable.

- UTMTAPECLEAN keeps track of clean dates and number of accesses since last clean of the tape.

- The 'free code' shows the method of how to release used but expired tapes for further write operations.

With UTM we have made a step further to a fully automated EDP administration where operations and resources are handled by the computer itself. This part of 'office automation' has been ignored in the past but will get more and more importance due to the fact that DP centers are growing rapidly while the number of people running them becomes less and less. A good administration system running on the computer leads to more security which means not only 'function' and 'access' but also 'knowing what is going on'.

INTRODUCTION TO RELATIONAL DATABASE TECHNOLOGY FOR THE HP/3000

Bill Hasling
COMPUTER RESOURCES INCORPORATED
5333 Betsy Ross Drive
Santa Clara, CA 95054

Introduction

During the last several years, few topics have received as much attention in computer trade publications as relational database theory and technology. But few topics remain as confusing or misunderstood. The mystique surrounding relational technology has led many manufacturers to claim their products to be relational or relational-like. These claims have only added to the confusion.

Relational database management systems (DBMS) have been proclaimed to be powerful, mathematically elegant, even futuristic. The major scientific publictions about database sytems are each year increasingly devoted to discussing the current research problems in the context of the relational data model. Yet, despite all the research and development done in database theory during the last 10 years the HP/3000 community has been nearly completely ignored. Powerful relational database systems are appearing for the IBM environment ("System/38", "SEQUEL/DS", "QUERY BY EXAMPLE") and for the DEC environment ("ORACLE" and "INGRES"). However, few of the benefits of this research and development in database systems have been utilized in the HP/3000 environment.

This paper will discuss several common questions concerning relational technology, especially as it applies to the HP/3000 user. We will first define database systems, discuss the three major database models, and then compare these systems to the relational model. A thorough understanding of the differences among the existing data base models will help foster an appreciation for the simplicity and flexibility of the relational database design.

What is the Difference between a Data Model and a Database System?

A database management system is a tool for organizing and manipulating data within a computer. Almost all software performs some sort of data management function, but a DBMS provides a structure for the data organization that allows many unrelated users, or programs, to manipulate and use a common collection of data.

A data model is the method used by a DBMS to organize and retrieve the data. There are three basic categories of data models: Network, Hierarchical and Relational. A DBMS may combine features of all three or only use a

subset of one or more of the models. This may be a source of much confusion if a company claims that it sells a relational or relational-like DBMS when, actually it only has a few of the features associated with the relational data model. There can be good or poor implementations of any of the three data models. As described later, each of the models supplies different characteristic benefits and drawbacks to the database management system.

What is the Value of a DBMS?

When does a DBMS provide some value over simply putting data into a disk file and retrieving it? The major benefits of a database system are concurrent access of data by several users, reduced data redundancy, data independence, data security, and the availability of database utilities.

A DBMS allows different users to access the same data according to their own needs without requiring their own copy of the data or information database. The DBMS forces a standard on the representation of the data, and acts as a data independent window that can insulate users from other users needs for the same data. As an analogy, a DBMS is to data what an operating system is to the machine. It allows several users to concurrently access the same resources, checks for inconsistencies, applies security restrictions, maintains integrity and provides utilities to assist the user in effective use of the DBMS.

What Type of Data Can Be Represented in a DBMS?

This may seem to be a naive question at first. After all, you can put anything into a database, right? Actually, there are many types of data that are difficult to represent even using state-of-the-art DBMS concepts. Currently, database systems store information as entities and relationships between entities. Pictures, abstract ideas or spacial information cannot yet be stored using typical database systems. An entity is an object represented in the database -- usually a record of data: a part, supplier, project, person, department, etc.

A relationship is a logical connection between two or more entities; for example, the fact that supplier SMITH supplies part FRAME, or person JOHN belongs to the department SALES. It turns out that this structure is sufficient for most business problems. Therefore, we will not concern ourselves here about trying to represent abstract data.

Both entities and relationships can be considered facts known by the database system. Entities may contain many associated pieces of information. A part may have a name, number, description, cost. Entities are straightforward and are represented in all three data models as files. The associated pieces of information of an entity are fields in a file. (For the remainder of this paper, entities and files will be used interchangeably).

All three data models represent entities in the same fashion. Relationships, on the other hand, are represented differently in each. The different methods force restrictions as to the type of relationships that can be represented, and the way data can be retrieved.

Relationships may be one-to-one, many-to-one or many-to-many. A many-to-one relationship is the most common, such as the relationship

between person and department. A single department may contain several people, but a person may belong to only one department. An example of a many-to-many relationship would be the relationship between students and courses. A single student may be enrolled in several courses and a single course usually contains many students. Relationships may also involve more than two entities, as the relationship between parts, suppliers and projects. A single supplier may supply many parts to many projects.

As the relationships between the data increase in number, the database becomes very complex. The method that the DBMS uses to represent the relationships and to manipulate the data affects the productivity of the end user. We will discuss how each of the three data models represent relationships and why some are superior different kinds of relationships.

What are the Characteristics of each of the Three Data Models?

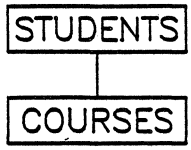
To answer this question, we will use a simple database, university enrollment, as an example and represent this database using each data model. We will then discuss the features associated with each model and its advantages and disadvantages. The database contains students, courses, and a relationship between students and courses (which students are taking which coures). In the example, students have a name and an age, and courses have a name and a room number.

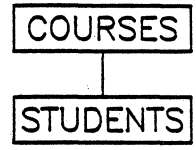
Hierarchical Data Model

The oldest of the data models is the hierarchical model. Given the problem of representing relationships between entities, the hierarchical approach is the most straightforward. Data is organized as in a file box or on a paper listing. The hierarchical model is characterized by a hierarchical ordering of the files. This ordering determines the relationship between the files and the preferred access path for finding the data. The hierarchical diagram (see Figure 1) shows the two different ways of organizing our database: one is grouped by students, the other is grouped by courses. Searching a hierarchical database can only be efficiently performed from top to bottom. In the first case A, it is simple to find out what classes BOB is taking, but hard to find out who is taking PHYSICS. The opposite is true in the organization B; It is simple to find out who is taking ENGLISH, but hard to find out what classes BOB is enrolled in. With a hierarchical data model you cannot have it both ways. You must decide how you want to access your data and pay a large efficiency penalty for asking the wrong questions. The hierarchical model is a simple view of a database that works very well for some database problems, but lacks flexibility.

Data retrieval inflexibility is only one undesirable property of the hierarchical model. It also can be space-inefficient because of data redundancy. In example A, two students are enrolled in PHYSICS. All of the data associated with PHYSICS is duplicated in the BOB record and in the MARY record. Data redundancy is not only inefficient in terms of storage, but also allows the possibility of changing the room for PHYSICS in one record and not in the other. This can cause the database to become outdated and inconsistent.

HIERARCHICAL





BOB 22	PHYSICS	ROOM A
	MATH	ROOM B
	ENGLISH	ROOM C
FRED 21	ENGLISH	ROOM C
	PHILOSOPHY	ROOM D
MARY 27	PHYSICS	ROOM A
	ENGLISH	ROOM C

(A)

PHYSICS	ROOM A	BOB 22
		MARY 27
MATH	ROOM B	BOB 22
ENGLISH	ROOM C	BOB 22
		MARY 27
		FRED 21
PHILOSPHY	ROOM D	FRED 21

(B)

FIGURE 1

The data redundancy problem, shown in this example, exists because the student/course relationship is a many-to-many relationship. The hierarchical data model does not deal with many-to-many relationships very well; however, it works effectively on many-to-one relationships. Because the many-to-one relationship is the most common, the hierarchical data model is frequently used with success.

One of the main purposes of a DBMS is to allow many users and applications to view a database in different ways -- this requires flexibility. The hierarchical model does not provide this flexibility. For this reason, more flexible data models have evolved. These particular problems inspired the concepts for the netword data model.

Network Data Model

The network model offers several improvements over the hierarchical model. The network model solves the data redundancy problem and the problem of representing many-to-many relationships. In the network model each entity is represented as an independent file and the relationships between files is represented by links. Usually, each link is given a name a provides a connection between two files. In the network diagram (see figure 2), there are two links: the student/course link, and the course/student link. The links are implemented in a network database as pointers, hash tables, or indexes, but in all cases are specified when the database is designed and therefore cannot be easily changed.

The standard network model has several good features. First, data is not redundantly stored as in the hierarchical model. Second, data can be accessed efficiently in many different ways, limited only by the foresight of the database designer. The complete network model allows links between any files in the database. This allows any type of relationship to be easily represented within the model.

While the network model provides a vast improvement over the hierarchical model, there are several disadvantages. First, although any type of relationship can be represented, it is very difficult to add, change, or delete relationships from a network database. Also, it is difficult to add, change, or delete files from a database as there is a tight binding between files and relationships. The network model offers the power to represent many different types of relationships, but the tight coupling between relationships and the associated files requires a major operation to make even minor changes to the data structure. For applications that are very well understood when the database is designed and do not change over time, the network model is excellent.

The network model can be a burden when it is used for applications that are only partially understood when the database is designed, or for applications whose requirements change with time -- such as applications that start small and slowly grow.

The second major disadvantage of the network model is the fact that the database designer must name the links in the model and normally the names must be specified by the user when querying the database. This makes navigating the database complex and data dependent. To ask the simple question involving a relationship, "Find all students taking PHYSICS," requires deciding whether to use the student/course relationship or the course/student

NETWORK

STUDENT

BOB	22
FRED	21
MARY	27

STUDENT->COURSE

COURSE->STUDENT

COURSE

PHYSICS	ROOM A
MATH	ROOM B
ENGLISH	ROOM C
PHILOSOPHY	ROOM D

FIGURE 2

RELATIONAL

STUDENT

BOB	22
FRED	21
MARY	27

CLASS

BOB	PHYSICS
BOB	MATH
BOB	ENGLISH
FRED	ENGLISH
FRED	PHILOSOPHY
MARY	PHYSICS
MARY	ENGLISH

COURSE

PHYSICS	ROOM A
MATH	ROOM B
ENGLISH	ROOM C
PHILOSOPHY	ROOM D

FIGURE 3

relationship. No matter which choice is made, the network model requires a procedural description of how to navigate the database. The procedure to find all students taking PHYSICS is:

1. Find the PHYSICS record in the course file.
2. Follow the course/student link to the student record.
3. Print the student record.
4. Go to 2 until all students are found.
5. End.

A similar procedure can be written to answer the same question by following the student/course link. Usually, one of the available choices is more efficient and it is up to the database user to decide which one to choose.

You will notice that the above procedure looks much like a computer program. This is characteristic of the network model: The link pointers of the network model are crucial to the data represented in the database. Due to the procedural nature of queries using the network model, it is difficult to construct a high-level front-end command interpreter. The network model is more at home with record-level manipulation as opposed to those of a higher level, such as, "Give me records with this property...". The relational model solves these remaining difficulties.

Relational Data Model

A true relational DBMS is a system that uses the relational data model for representing relationships and the relational access method for retrieving data.

In a recent letter to Computerworld, E. F. Codd defined what makes a system relational:

> " A true relational system requires all information to be represented at the logical level as values in tables. There must be no user-visible navigation links between tables and the system must support (in some form) at least the select, project and join (natural or equijoin) operators of the relational algebra."

More simply, a relational system:

1. Is made up of only flat files, i.e., tables.
2. Allows the user to pick any fields from any file for data manipulation (project operator).
3. Allows the user to select any set of records for manipulation, subject to some condition (select operator).
4. Allows two or more files to be joined into one logical file where fields from records in each file match (join operator).

The relational data model has a strong mathematical basis. Literature on relational models discuss the select, project and join operators. These are the mathematical terms used to define the operators of the relational model. The join operator is the key to the relational model. As an example of a join operation, we can join the CLASS file and the COURSE file (see Figure 3) by matching the course-name field. The result would be a logical file containing the following data:

BOB	PHYSICS	ROOM A
BOB	MATH	ROOM B
BOB	ENGLISH	ROOM C
FRED	ENGLISH	ROOM C
FRED	PHILOSOPHY	ROOM D
MARY	PHYSICS	ROOM A
MARY	ENGLISH	ROOM C

Following the join operations, we can print all the classroom where BOB attends by further selecting only the records with the name BOB. By using the select, project and join operators, any arbitrarily complex question can be answered.

The relational model appears to be similar in some ways to the network model, except that relationships, represented in the network model by links, are represented in the relational model as another flat data file.

In our example, the new data file contains a field from each of the entity files (in the class file, student name and course name). Each record in this new file establishes a link between the two entity files. This new file is called a **relation.**

If we take a second look at figure 3, we can see that our new relation appears identical to the other entities in the database: They are all simply two-dimensional files with fields. The beauty of the relational model is that relationships are stored simply as another entity in the database. The link between the entities is logical, not physical. This loose coupling between files allows much greater freedom to add, delete and change the database structure. New files can be added and new relationships created without affecting the existing database. Changes can be made to the structure of a file without effecting the rest of the database. The database may be designed slowly, one file at a time or modified to change with a changing environment.

The relational model provides the same representational power as the network model, but provides additonal dynamic control over the database. It is simple to create new relationships or add new files to the database. This increases the flexibility of the system which in turn makes the relational model more powerful than either of the other two models.

The second major advantage of the relational model is that the user need not specify the most efficient path to find the data. In the network model, to find all the students taking physics we need to specify that we want to use the student/course link or the course/student link to join the files. In the relational model, we specify that the class file is to be joined with the student and course files and leave the job of deciding how best to perform the join to the DBMS. This capability allows the user to issue high level queries. For example, the high level command to find all students taking physics would be:

```
SELECT STUDENT WHERE CNAME="PHYSICS" AND
    CLASS.SNAME=STUDENT.SNAME AND
    CLASS.CNAME=COURSE.CNAME
```

The construct CLASS.SNAME refers to the SNAME field in the CLASS file. The two conditions

```
CLASS.SNAME=STUDENT.SNAME and
CLASS.CNAME=COURSE.CNAME
```

are called join conditions and specify how the files should be logically linked. The join conditions can be made invisible to the end user or manually applied as in this example. This high-level query is nonprocedural. In relational terminology the query creates a set, which is also called a view, or a virtual file. This virtual file is itself a relation, illustrating again the symmetry of the relational model. We can perform nearly every operation to a virtual file as we can to a real file.

Because data is stored in familiar, two-dimensional tables, users can create their own files and manipulate data through simple, interactive queries using an English-like, nonprocedural command language. Because of their simple design, relational databases are easy to modify and can grow as the application grows. The relational model is understandable even to the casual computer user.

Are Relational DBMS's Compatible With Other Data Models?

One of the greatest concerns of computer users is compatibility with existing software and data. Users are always concerned about introducing new software into their environment that may require changing existing files or programs, and retraining personnel. Since the relational model handles only flat files similar to those used by he network and hierarchical data models, there is no reason that a relational DBMS could not be used with files created by another DBMS which used a different data model. In fact, RELATE/3000, the relational DBMS available for the HP/3000, can access IMAGE databases as well as KSAM and MPE files, the other HP/3000 databases. The capability to interface to several file types allows organizations to gradually convert to relational technology and to use some of the power of the relational access method with databases from the older data models.

Are Relational DBMS's Efficient?

A common misconception is that relational DBMS are slow and inefficient. This undeserved reputation has been gained from two sources. First, in the early days of relational technology, two major research organizations were attempting to show the feasibility of the relational data model. The two projects, INGRES, at University of California, Berkeley, and SYSTEM R, at IBM, were both research vehicles designed to show that a working database system could be built using the relational model. Both of these research attempts were successful. However, the implementations were not highly efficient because they were designed as research tools, not as production systems. Relational database systems are just as efficient as network systems and can be as efficient as the hierarchical model.

The second reason relational database systems have sometimes been perceived as slow is that the flexibility of the relational model allows the user to construct queries that may require extremely difficult database manipulations. These manipulations may be time consuming and, consequently, give the database the appearance of being slow. Yet these same tasks are ones that cannot be done at all with conventional query languages. It is far better that a vital task be done

slowly than not done at all.

How Does a Relational DBMS Efficiently Access Data?

A detailed answer to this question is beyond the scope of this paper, but a brief answer may suffice. Many different techniques have been developed to efficiently access relational databases. These techniques ask the DBMS to analyze the query, examine the database structure, and to generate an efficient procedure to access the data. The major reason relational database systems can be efficient is the intelligence that is built in to the access mechanism.

One or more secondary indexes are usually associated with each file in a relational database system. An index is a mechanism that can be used to quickly find a particular record in a file, given some combination of fields from the file. Common indexing schemes are hashing functions, B-TREE indexes or sorted keys. The intelligent relational access method uses these indexes to reduce the search for records in the database. The indexes play a role similar to the links in the network model, but the indexes are associated with a file and not a pair of files. The indexes may be created or deleted at any time. They may effect the processing speed, but do not affect the data or relationships between the data.

How Can Relational Technology Help Solve Buisness Problems?

This is the ultimate question, because this is what users want most from a database system. The system designer may be impressed with the symmetry, generality and the mathematical beauty of the relational model, but the end user just wants to get his data in and out the way he or she wants it, as fast as possible, with a minimum of difficulty. The major advantages of the relational model are:

1) The power to represent all common relationships as they are seen by the user, without distorting the relationships to fit the data model.

2) The flexibility to easily, and quickly, change the structure of the database to reflect changing user needs. These changes often have little or no impact on applications that access the data.

3) The ability to efficiently access the data in many different ways without requiring all possiblities to be designed into the database in advance.

4) The ability to query the database in a non-procedural fashion makes it possible to develop very high level query languages that an inexperienced user can use to view the data as desired.

5) The flat nature of relational files allows a gradual conversion to relational technology thereby minimizing the impact on existing systems and applications.

Relational technology is the direction of the future in database systems. Information about new relational database systems is appearing more often in trade journals and vendor announcements. Relational DBMS systems can save hours of time when making simple changes to databases and make many applications possible that were too complicated, time consuming, or impractical to do with previous technology. Relational DBMS are no longer the wave of the future -- they are here now to solve today's business needs.

References:

Articles on Relational Database Systems:

Astrahan, Blasgen, Chamberlin, Fswaran, Gray, Griffiths, King, Lorie, McJones, Mehl, Putzolu, Traiger, Wade, Watson. "System R: Relational Approach to Database Management.". ACM Transactions on Database Systems. June 1976, pp. 97-137.

Brodie, Schmidt. "Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group." ACM SIGMOD Record, July 1982.

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." CACM, no. 13,(1970), p. 377-387.

Codd, E.F. "Relational Database: A Practical Foundation for Productivity." CACM, vol. 25 no. 2, February 1982, pp. 109-117.

E.F. Codd, "Letters," Computerworld, June 7, 1982, p. 64.

Kreps, Stonebraker, Wong. "The Design and Implementation of INGRES." ACM Transactions on Database Systems, September 1976, pp. 189-222. Wong, Youssefi. "Decomposition -- A Strategy for Query Processing." ACM Transactions on Database Systems, no. 1,3, September 1976, pp. 223-241.

Books and Periodical Issues Devoted to Database Systems:

ACM Computing Surveys, Vol. 8 no. 1, March 1976.

Date, C.J. An Introduction to Database Systems. Addison-Wesley. 1977.

Datamation, September 1981.

BETTER BUSINESS PLANNING THROUGH SYSTEM INTEGRATION

Rune Haugsoen, Financials Product Manager

ASK Computer Systems, Inc.

Introduction

The proliferation of interactive mini-computers over the last decade and the emergence of a new decade of micro-computers have greatly enhanced the use of planning systems in operating a business. On the software side, the last decade generated a number of comprehensive modeling systems. Initially, these were designed for mainframes but are now also becoming readily available on minis. Along the wave of the personal computers, professional planning software is evolving at an accelerated rate.

This paper addresses those aspects of a typical business where computerized planning can be utilized, and also how the various categories of computerized planning tools can be integrated into one cohesive Decision Support System. The discussion will focus on the needs of a company operating in a manufacturing environment. The size of the business is perceived to be a small company or a major division within a corporation.

Situation

In order to effectively run their businesses, most companies are engaged in planning in three essential areas. The first is demand management, which involves the sales organization, order management, and the billing function. The second area is resource management, whose primary objective is to meet market demand. This is typically done through engineering, material planning and production. Third, the company's activities are translated into the common denominator, of monetary value, and are planned, tracked and analyzed by accounting.

Planning is done on several levels. For the intent of this paper, we will distinguish between strategic, tactical and operational planning. Strategic planning states the overall objective and direction of the company. Tactical planning outlines solutions that will implement the strategic plan while the operational planning details the

actual execution of the two above levels of planning. Ideally, the three levels of planning should tie together in a congruent whole.

Different parts of the organization are engaged in different levels of planning. Strategic planning is typically carried out on a corporate or divisional level. Tactical planning is handled by divisions and departments and the various subdepartments will do the operational planning.

Requirements to planning systems are vastly different depending on which planning level they focus. The first levels of management are mainly engaged in using the operational systems for their planning needs. A typical example of this is the Material Requirements Planning System (MRP). Characteristics of these operational systems are a focus on high processing efficiency, control of input data for the insurance of overall information validity and the requirements to the system's output tends to be rather static.

The middle level management's focus is to devise plans that will meet the company's objectives. In order to accomplish this, they need planning tools where data can be manipulated. Traditionally, this function was done manually through various spreadsheets. Several iterations would have to occur, including laborous recalculation of numbers, before an acceptable plan was developed. Financial budgeting is one of the best examples of this process. Manual recalculation of spreadsheets are no longer necessary. VISICALC - the interactive data manipulating tool currently marketed by VISICORP - has automated this function and, thereby, also revolutionized the professional executive's use of micro computers.

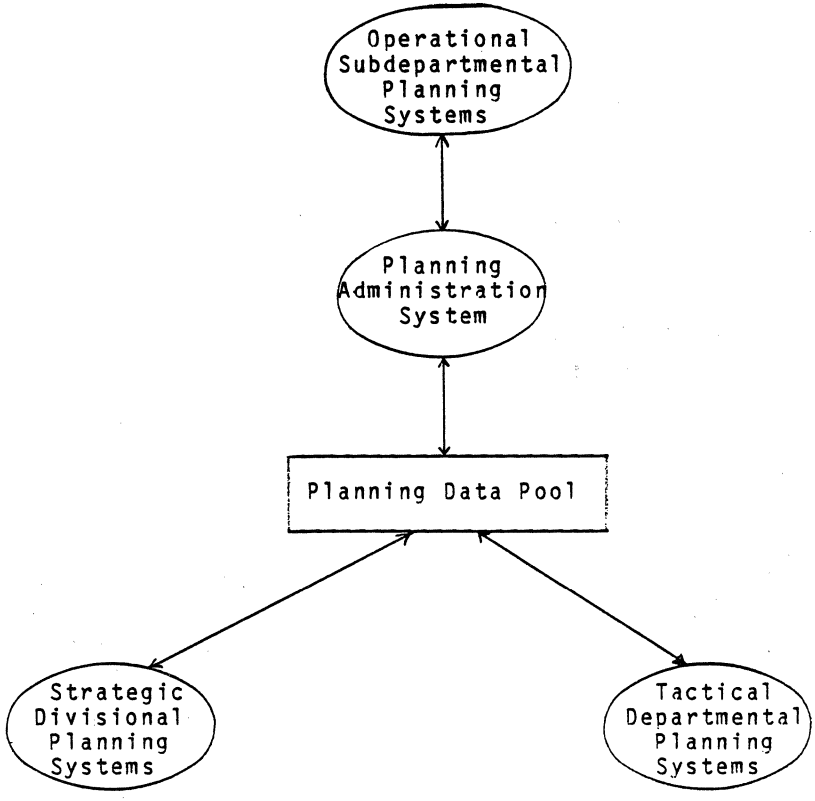
Higher levels of management, who define the overall objectives and who also analyze the impact of various business strategies, need to develop models that emulate particular business situations. Model parameters need to be changed easily in order to do "What - if?" analysis, simulation and goal seeking. Modeling systems are becoming increasingly available on mini computers. A key characteristic of these systems is ease and flexibility in specifying and changing the model's processing of data.

With the emergence of these planning systems, namely, the professional and modeling applications, a new concept called Decision Support Systems (DSS) is being formed. The essential characteristic of DSS is that the impact of various decisions can be calculated and analyzed with the formation of a model and with the inclusion of various business assumptions. These assumptions could be a number of items, e.g. people count, production level, or salary

estimates, and the model could range anything from a budget to a comprehensive statistical model. Thus, the concept of DSS comprises both tactical and strategical planning tools. These facilities can be used directly by the professional or management end user. Now that adequate tools are available, the challenge is to define the decision processes that can be supported by systems, select and develop the appropriate tools, and integrate this into one cohesive solution.

Integration of Planning Systems

To ensure integration and consistency of plans on the various levels it is important to standardize the format and to provide the needed interfaces. The diagram below describes, conceptually, how the numerous planning functions could tie together.



The objectives of this solution are to make planning more flexible, to enable a structuring of the planning process and to protect integrity of the planning data. In order to achieve this, consideration needs to be given to the human, software, and hardware elements operating in this environment.

On the human side, one can distinguish among the people who own the operational systems, those who generate the plans and the individuals who are responsible for the functionality of the systems. The planning process can be illustrated by considering how financial budgeting is done. The ownership of the financial systems lies with the controller's department. This department will set up initial worksheets for the other departments involved in planning and will also possibly provide historical information as a base for budgeting or generate overall targets for the planning period. These will be submitted to the involved departments or responsibility centers so that initial budgeting can begin. First-pass budgets from the various departments will be consolidated and analyzed with regard to the overall business results. Several iterations will most likely take place before satisfactory budgets have been developed and input to the accounting systems. At the point of update, the controller will be concerned that the budgets are valid. Checks should be made so that the correct budget and fiscal periods are affected. Editing needs to be done with regard to such items as account number, magnitude of budget and scaling of amounts.

Although this discussion focuses on financial budgeting , the same process can be generalized to product sales forcasting and production planning. The departmental plans can then be tied into an overall business plan. The monitoring of the planning process as well as the building of models will, in many cases, be done by a system administrator/user analyst who has the necessary technical knowledge and who also has this area assigned as his/her responsibility.

With the current state-of-the-art technology, one could argue that two types of hardware are needed. One is the micro computer, where the plans will reside and be analyzed. The argument for this is that the distribution of processing to a micro is advantageous in terms of response time when heavy data manipulation is needed. On the other hand, processing of the operational systems as well as processing the more complex models require the data processing power of a mini-computer.

Consequently, the mini will contain the operational system with its databases; the modeling system, including models and related database; a Planning Administration System

(PAS) that will manage the extraction and updating of planning information to the operational system, and a planning data pool where formatted planning data can be exchanged between the different planning modules. The micro will store the data communication software needed to communicate with the host (mini) computer. In addition, it will keep the professional planning systems and also store the information local to particular department's planning.

Planned data can be transferred between the planning data pool and any of the three planning modules. This implies that besides interfaces between the operational, tactical and strategical planning modules, one can also traverse planning systems within the same module. An example of this would be transferring information from one personal computer to a personal computer of a different make.

The Planning Administration System is, in essence, the hub of the Decision Support System function. In general, PAS will perform three functions: it will extract information from the operational system; it will format data to comply with the format of a particular planning system, e.g. format information to a VISICALC readable matrix; and it will update planning data to the operational system.

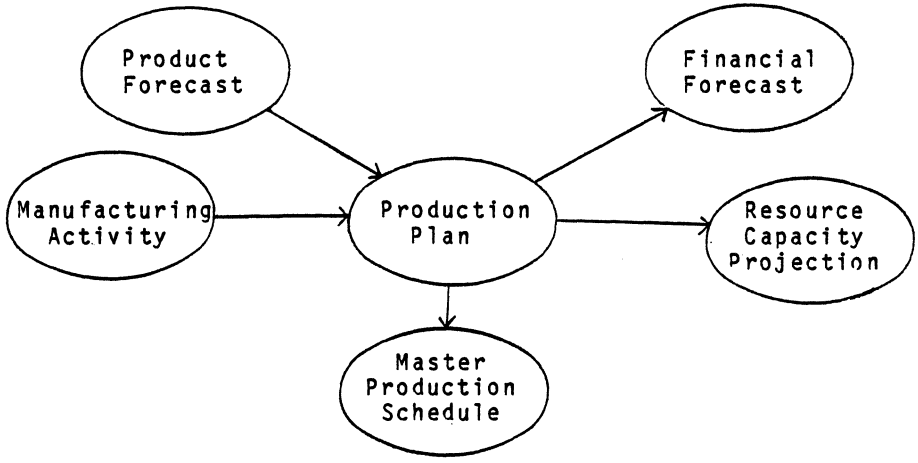
The extraction function can be generalized such that the user can define any information that needs to be captured from the operational database. Obviously, such a function would need to be flexible and would basically resemble the features of a QUERY language. The extraction features could also be predefined within PAS. Particularly in the area of budgeting, product forecasting and production planning it would be reasonable to predefine a standardized format. The update function to the operational system will always be of a standardized nature to secure system integrity. The focus of the updates will be to the financial, demand, and resource management sides of the operational systems. Data security needs to be considered. PAS will use passwords and conventions from each related operational system. In this manner, not only will unwanted access to the information be prevented, but it will also enable the owner of the operational system to direct the planning process. The controller can use PAS to set up departmental worksheets for budgeting and make them available to the departments in the planning data pool. The departments will then use DSS tools to actually generate the budgets that will be updated back to the planning data pool. It will then be up to the controller's discretion whether he wants to input the budgets to the operational system, do further analysis, or reject them.

The flexibility of extraction of any operational data will enhance not only the ability to do comprehensive analysis on a personal computer, but also the operational data could

form the basis for producing departmental plans for a subsequent update back to the operational system. To use an example, let's consider again a financial worksheet for a department's budget which could be predefined in PAS. The department manager would see account numbers, account description and possibly last years actual figures for an appropriate planning horizon. If the budgets were flexible with regard to actual production activity, then the labor hours could be captured from the operational system and submitted to a personal computer. This information could then be combined with related algorithms and standard labor rates to form input to the flexible budget for a given fiscal period.

Why would a separate modeling system be needed and what are it's characteristics? A modeling system, or more correctly, a modeling language is basically a high-level programming language especially aimed at the building of models. Several special features are normally included; for example, a submodule for financial anaylsis or statistical analysis. There are several reasons for including a separate modeling system. First it would be useful for more complex modeling. A representative example of this would be optimization models, or the simulating of a business situation using probability distributions with the need for numerous iterations. Another example could be product forecasting where the computer would use alternative statistical methods for the forecast of a large number of products. In these situations limitations of a VISICALC-like product would become very apparent. Secondly, a modeling system would be an invaluable tool in consolidating the various departmental plans into an overall business plan. The focus could be to obtain visibility of the overall business impact of the various departmental plans before they are submitted to the operational system.

The exhibit below gives the overview of how, conceptually, such a business planning system could tie together.



Product forecasting would be developed by the sales and marketing departments. To aid the planning process, an initial product forecast could be generated by a forecasting model using mainly historical data. This would then be submitted to the product planner who would revise the numbers by using a VISICALC-like planning tool. The final demand situation could then be input into a Production Planning model, where the demand would be transformed into an actual production commitment. The production plan could then be tested against a resource capacity projection to check validity of the production rates. Also, feedback on actual production activity could be made available for the production planning, giving current inventory levels and shipment history as a basis for future planning. Alternative production plans could form input to the financial planning module to analyze the overall fiscal consequence of each particular plan. The final production plan could then form input to the master production schedule for further operational planning and control.

Conclusion

This paper has aimed to conceptualize how categories of planning modules can tie together into an overall solution. The focus has been on the vertical planning process between divisions, departments and subdepartments. The horizontal communication, which is often of a less formal nature, is equally important and is addressed in papers dealing with office automation.

The last few years have brought forth tools that significantly will enhance a company's planning activities. The initial focal point in a company's selection of DSS

tools will be automation. Lack of overall planning and coordination will most likely result in each professional choosing, independently, his/her own personal micro with the resulting increase of the individual's productivity.

The synergetic effect, however, from the use of DSS tools can be an increase in productivity coming from better organizational communication as it relates to formal planning. To achieve this, long-range planning is needed to create a conceptual framework for directing the process of implementing DSS.

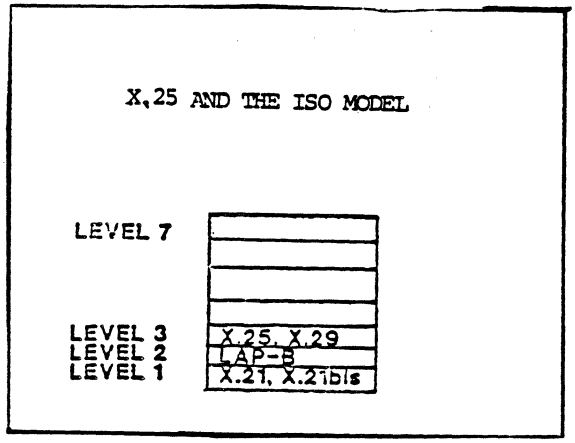
USING X.25 COMMUNICATIONS FOR NETWORKING APPLICATIONS

Suzanne Hernandez

Hewlett-Packard
Information Networks Division

As with most standards, the name X.25 gives little clue to the capability referenced. The X.25 standard is not really a standard, but rather a recommendation of the International Consultative Committee for Telephone and Telegraph (CCITT), the latest version of which is October 1980.

We will relate the X.25 standard to another international datacommunications standard which you may be familiar with, the Open Systems Interconnect Model released by the International Standards Organization. This is a 7-level model for communications between systems or devices. The levels range from 1 to 7.



The world of datacommunications is complex, and the OSI model attempts to add some order by defining distinct layers and interfaces between these layers. This architecture serves as a model for evolving and future datacommunications products. The lowest level corresponds to the PHYSICAL communications level, covering voltage levels and physical connections. Two examples of level 1 standards are RS-232-C and RS-449. As you may have guessed, moving up the levels leads to higher levels of complexity. Level 2 is the LINK level and corresponds to the low level protocol for reliable transmission of data across the medium, examples of which are HDLC, Bisync, and SDLC. Level 3 is the NETWORK level which provides the ability to establish communications between two different networks. The fourth level, called the TRANSPORT layer, provides the end-to-end integrity between the systems and processes with the services provided to the upper layers (5-7) independent of the underlaying network implementation (1-4). Layer 5, the SESSION layer supports the dialog between cooperating entities, binding and unbinding them in a communicating relationship, establishing a communications "session". The PRESENTATION layer is what most systems programmers desire but rarely

receive. This layer provides the services to allow the application process or user to interpret the data in a meaningful way, performing such tasks as data format, file, and data base conversion. The uppermost layer, the APPLICATION layer, directly serves the end user by providing the distributed information service to support the application process, application management, and the systems management. To put this in perspective, DSN/DS for the HP3000 provides capabilities which span all 7 communications levels indicated above.

But we are discussing X.25 so lets see how this fits in. The X.25 recommendation conforms to the lowest three level of the OSI model. The physical layer references recommendation X.21 and X.21 bis. X.21 defines the physical interface (15 pin connector) and the procedures for establishing connections through circuit switched networks. The majority of the datacommunications equipment in the world today does not implement the X.21 standard. To address the reality of the situation, the CCITT established X.21 bis as an interim standard. This standard conforms to RS-232-C, a standard which all manufacturers and users are familiar with.

The packet level of X.25 maps closely into level 3 of the X.25 recommendation. Within the packet level two types of communication are supported, Virtual Circuits and Datagrams. A Virtual Circuit is established between the two end points. This means that there is a logical connection between the two entities, and at different times the data may travel over different paths between the end points, but to the user it appears as a single "wire" between the points. One form of virtual circuit is a Permanent Virtual Circuit which is similar to a leased phone line (i.e. it is continuously established). The second form is a Virtual Call where the virtual circuit is only established for a period of time and then torn down, similar to a dial-up phone line connecton. Both of these virtual circuits present data "packets" at the destination end in the same order as they were sent from the source. The Datagram service provides for each packet to be treated independently as an individual message with no guarantee of data sequence at the destination end (This service is not available with the X.25 capability for the HP3000.)

The packet level takes the information from the upper communication layers (4-7) and forms a packet with two basic parts, peer-to-peer protocol information in the header and user data. It is important to realize that X.25 does not provide communications capabilities similar to remote file transfer, virtual terminal, and program to program communication as DSN/DS.

We have discussed some of the technical aspects of X.25, but now lets discuss what it really provides the user.

A Packet Switched Network is a communications network often shown as a "cloud" with computers and terminals connected. The packet switched network is a communications network formed of "switching nodes" and "trunk lines". The switching nodes route traffic throughout the network from source to destination over the trunk lines which can be either 56kbps digital phone lines, satellite links, or microwave links. The type of transmission medium and protocal utilized within the network is irrelevant to the network user. X.25 defines the interface between the DTE (computer or terminal) and the DCE (X.25 network). The network may actually use an X.25 protocol within the network or it may be a propriatory communication scheme, but in either case, the network user does not care.

The Packet Switched Network (PSN) allows for the connection of both computers and terminals. For computers to connect via the X.25 communications protocol, all three layers including physical, protocol, and packet must be implemented on the computer. The geographic nature of the PSN makes it well suited as a means for connecting a large number of remote terminals to one or many computer systems. Since most terminals provide only a RS-232-C ASCII capability, a protocol has been established to allow these terminals to communicate with a host computer connected to the network. Via a full duplex modem the terminal connects to the network and communicates with a function called a Packet Assembler-Dissassembler or PAD. This function takes the asynchronous characters transmitted by the terminal, forms them into packets and transmits them to the computer. Related to the terminal communications, you may hear the standards X.3,X.28, and X.29 which are related to X.25. The PAD function is described by the X.3 recommendation, X.28 describes the protocol between the terminal and the PAD, and X.29 describes the protocol between the computer and the PAD, and lies logically above the packet level of X.25.

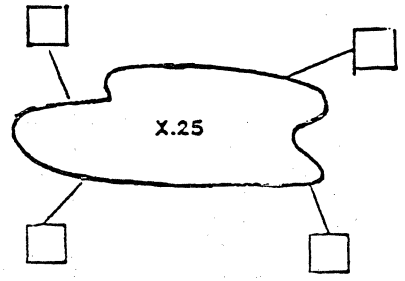
DSN/ X.25 PROVIDES

Access to X.25 Packet Switched Networks (PSNs)

and

Standard for Interfacing to non-DP Equipment

ACCESS TO PSN's



PSN's ARE FOR CUSTOMERS

WITH:

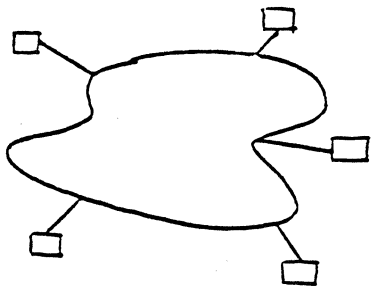
Geographically Dispersed Networks

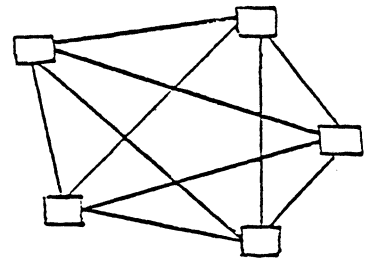
OR

Medium Volumes of Data

OR

High Reliability Needs





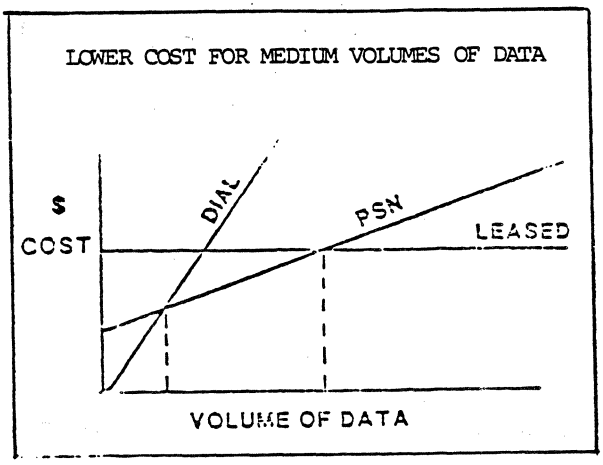
Via X.25 PSN: 5 Interfaces [= number of nodes, n]

Via point-to-point: 10 Interfaces [= (n-1)+(n-2)+...1]

The Virtual Circuit capability allows communication with many devices over a single interface, reducing the interface card requirements.

The ability to communicate between systems and from terminals to systems is currently available with dial-up and leased phone lines so what advantage do PSN's offer? PSNs provide advantages in three areas, first for customers who have large geographically dispersed networks, secondly for those having medium volumes of data to transfer, and thirdly for those requiring high reliability.

As we can see on the previous slide, with one connection to the network, a computer or terminal can communicate to any other computer or terminal connected to the network. This can result in a dramatic reduction in the required interfaces over a point-to-point full mesh network as shown below for 5 computers:



The second benefit of PSN's is lower cost for medium volumes of data transferred. This savings is dependent upon the tariff structure of the particular network and the relationship of these costs to the standard leased and dial-up phone lines. The example shown on the above slide is a generalized case for communications between two nodes. In a real network application, an X.25 network would in most cases not be used for a two node network.

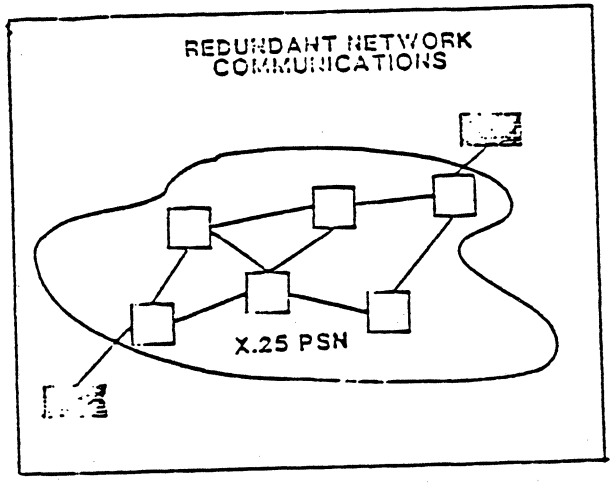
The graph shows communication cost as a function of the volume of data transmitted. With dial-up lines the communication cost is proportional to the amount of data transmitted, since this is usually roughly proportional to the connect time. For the leased line case, a fixed charge is paid regardless of the amount of data transferred and is shown by the horizontal line. The tariff structure of the Packet Switched Network is a combination of the dial-up and leased line structures. For connection to a PSN there is an initial installation charge, after which the user pays a fixed monthly charge plus a certain amount on a per packet basis. You can see that depending on the various tariff rates, the PSN can provide a lower cost solution for medium volumes of data.

What is not shown in the slide is the effect upon the structure of adding additional nodes to the network which must communicate fully among themselves. For each additional direct connection to another node, the full fixed charge of a leased line must be incurred, whereas for the PSN connection only the smaller fixed charge is added, making PSNs more attractive as the number of interconnected nodes increases.

Another variable which is not currently shown in the slide is the distance between nodes. For dial-up and leased lines the charges increase with distance (e.g. a leased line from NY to Los Angeles costs more than a leased line from San Francisco to Los Angeles). With PSN's however there is a difference, with the connect charge and per packet charge cost being independent of the distance between nodes connected to the network. This situation serves to make the PSN more attractive on a cost basis as the distance between nodes increases.

As you can see there are many variables which must be considered when evaluating the cost of establishing a remote communications network utilizing X.25 PSNs against leased and dial-up lines. In summary some of these are:

*Number of nodes
*Volume of data transmitted
*Distance between nodes
*Pervasiveness for need to communicate between all nodes



The third inherent benefit of X.25 networks is that their internal network structure provides for alternate communication paths between switching nodes. Should an internal node or communication line fail, the PSN will automatically reroute to an alternate path, similar to the automatic rerouting of DSN/DS for the HP1000. The redundant capability is offset somewhat by the additional components and their inherent failure rates.

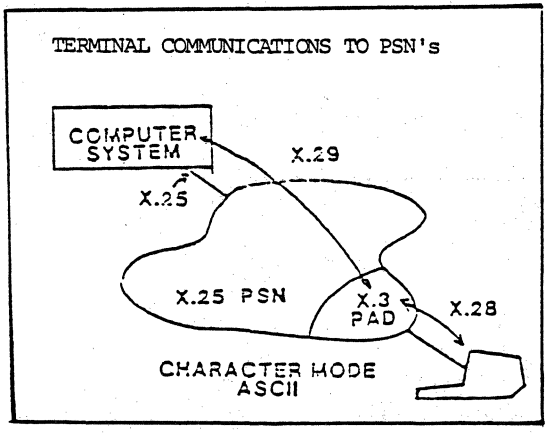
A point to note here is that the PSN may use the X.25 communication standard for communication between it's own switching nodes or it may not. From a network user's standpoint this is irrelevant since the user is only concerned that the interface between the device and the network (DTE-DCE) conforms to the X.25 standard.

Most countries have installed or are planning to install an X.25 Packet Switched Network. Within the US, the networks belong to private companies which lease phone lines from Bell to implement the switching network.

In most other foreign countries, the common carrier facilities are owned ad operated by the Postal, Telephone, and Telegraph (PTT) which is part of the national government.

The networks which the HP3000 currently supports are Telnet(US), Tymnet(US), Transpac(France) and Datapac(Canada).

Until now we have been discussing only system to system communications capabilities of X.25 PSNs. There is also the ability for terminals to connect to systems over PSN. Since most terminals are non-intelligent devices, capable of asychronous ASCII communications only, they cannot communicate over the PSN in the same way systems do via the X.25 protocol.



The PSN benefit of allowing many geographical loctions to connect to each other with a single connection makes it ideally suited for terminal networks which must communicate with a number of host computers remotely. To provide this communications, the PSN provides a Packet Asembler-Dissassembler (PAD) function to which the terminal can communicate. This PAD is defined by the X.3 standard. The function of the PAD is just as the name implies. The PAD takes asychronous ASCII from the terminal, assembles it into X.25 packets, and sends it to the host computer which then disassembles the packet and presents the ASCII data stream

to the operating system. The PAD has a number of variables which can be tailored to the desired communication between the terminal and the computer. The protocol between the terminal and the PAD is referred to as X.28 and the protocol between the PAD and the host computer is X.29. For the HP3000 systems, once the connection to the PAD and then to the system is established, the terminal functions as a standard MPE system terminal. There are restrictions with PAD use such as no block mode transfers and no binary data PAD transfers. With no flow control it would be possible to overflow the buffers in a block node situation and binary data transferred to the terminal could be interperted as control commands for the PAD and have unpredictable results.

While we are on the topic of other standards, there is another standard which is occasionally mentioned, X.75. This standard is not accessed by a user, but rather defines a protocol and provides a gateway function between networks such that a computer system connected to the Telenet PSN in the US can communicate with a computer connected to the Transpac PSN in France. This inter-PSN capability is possible technically, but is sometimes not available for tariff reasons. With international communications there is often the question of where the tariff charges are billed to, and if there are no agreements between the networks, then communications between PSN's is prohibited.

(This space left intentionally blank)

STANDARD INTERFACE
FOR CUSTOMERS
WANTING TO:

Interface to Non-HP Equipment

OR

Developing their own Upper-Level Networking (HP 1000 ONLY)

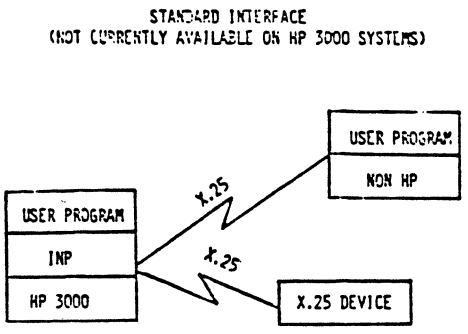
We have just covered the first benefit of X.25 as a standard for interfacing to PSNs. Now we will discuss the second possible use of X.25, as a standard for either interfacing to non-HP equipment or developing custom upper layer networking software. Since X.25 is a standard, we would assume that if two computers or devices both provided the same standard, that they could communicate via this standard. While this is generally true for X.25, there are some facts which you should be aware of.

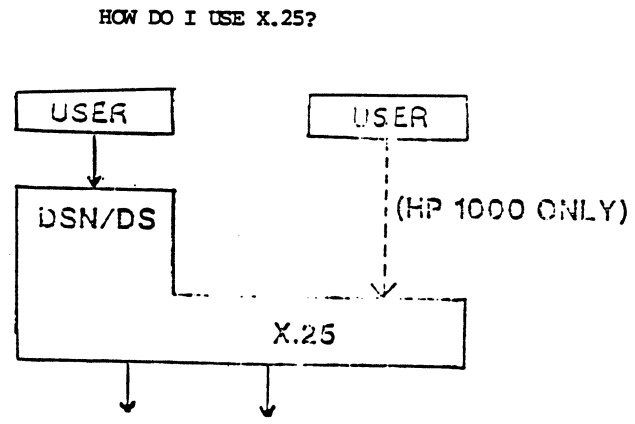
1. The X.25 product for the HP3000 family does not provide direct program access to level 3. Use with DS/3000 to other systems with DS/3000 is the only supported connection. The HP 1000 X.25 product however, does provide programmatic access to X.25 level 3. To clarify the distinction we will discuss the operation and benefits of a level 3 access capability.

2. When connecting to a X.25 PSN, the PSN always functions as a DCE while the computer functions as a DTE. If we were to connect two computers directly to one another, then one must be able to function as a DCE. The majority of X.25 implementations, HP3000 included, provide this capability but in general this must be checked.

3. All standards are not identical! This may sound a bit contradictory, but it is possible for two computers to adhere to the X.25 standard and yet not be able to communicate. There are certain parameters and sections within the specification where a choice can be made by the implementer, and one of many paths can be choosen. In the ideal case, the multiple choices can be implemented and software switchable to provide the desired configuration as needed. However, not all implementations are this flexible, and it is important to obtain the specification of the various implementations and compare them for differences to ensure that the connection will function correctly.

Even though two X.25 implementations on dissimilar computers can communicate, remember that X.25 only provides the lower three layers of the ISO model, not full networking. To provide meaningful data transfers between computers over X.25, user programs must be written to pass data buffers, perform file format conversion, perform data and floating point format conversion, provide high level end-to-end reliability etc.





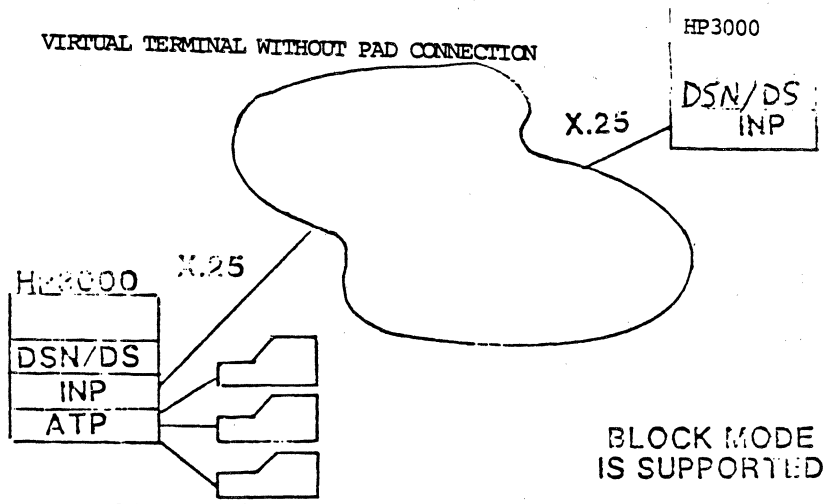
As you may have guessed it would be extremely desirable if the computer vendor would provide the upper level software for communication over X.25 rather than forcing the user to write this software. We as you can see in the slide, DS/3000 provides this upper layer capatibility.

For the HP3000, DSN/DS software is utilized over the X.25 protocol to provide the upper layer functionality. With this combination the full features of DSN/DS such as Remote File Transfer, Virtual Terminal, Program to Program Communication, Remote Data Base Access at the application layer are available over X.25 PSNs.

A DSN/DS user can communicate within a 3000 network containing local and remote nodes connected via point-to-point, or through an X.25 PSN, unaware of the particular transport mechanism utilized.

When two HP3000 computers are connected, using DSN/X.25 to make the DS connection, terminals connected to the remote HP3000 can communicate via the virtual terminal feature of DS/3000. In this configuration block mode is supported. DS is using X.25 as the transport mechanism. In order to implement this configuration only DS/3000 software with X.25 support is required, along with the INP for the X.25 link, and the ATP (or another appropriate) terminal controller. The important point to be made her is that there are no differences in the features of DS/3000 when using X.25 for a transport.

In the next slide, there is a configuration in which terminals are connected to the HP3000 via an X.25 PSN. In this configuration, block mode is not supported for the same reasons that were discussed previously, namely that there is limited flow control between the block mode terminal and the PAD. Therefore, there is potential for overunning the PAD buffers.

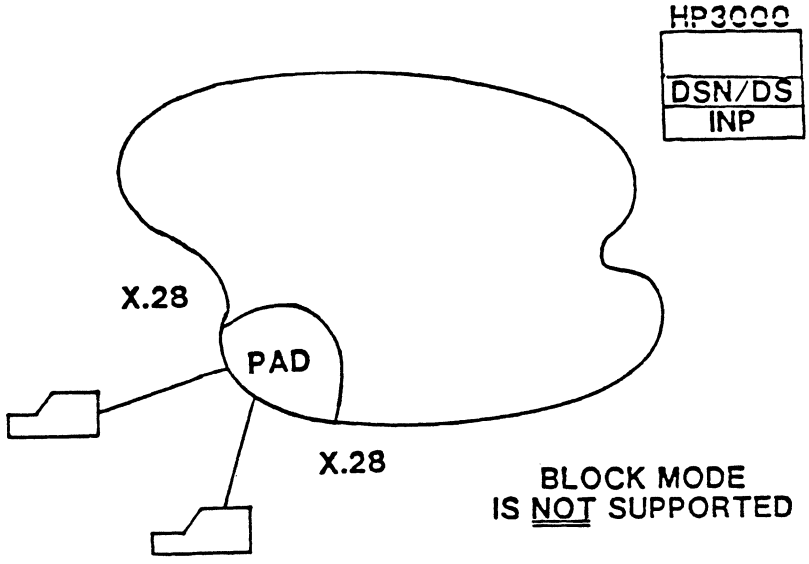


A distinction should be made here between the HP3000 and HP1000 implementation of the X.25 capability. For the HP3000, DSN/X.25 is part of DSN/DS, no separate product is required, but there is also no upper level interface at level 3. In the HP1000 implementation, DSN/X.25 is a separate product with a user interface to level 3. For terminal communications to a HP3000 over X.25 PSNs, DSN/DS must be purchased, and then will function as a standard MPE terminal (block mode is not supported over the PSN).

As indicated, multiple interface cards are supported in a single system, providing multiple X.25 links to the computer.

We have covered most of the areas of X.25, so now lets summarize the main points. X.25 provides two new capabilities for the HP3000 computer, first it provides access to X.25 PSN, and secondly it is an international standard interface.

The HP3000 product implementation has two advantages, the first is that all the DSN/DS features are available when the X.25 PSN is used for system to system communications. Secondly, terminals can access the HP3000 system via a PSN as long as it is in character mode only. The addition of X.25 greatly expands the communication capabilities of the HP3000 computer family.



POINTS TO REMEMBER

X.25 Provides:

- o Access to X.25 PSNs
- o Standard Interface

HP Advantages:

- o Use with DSN/DS

SELECTION CRITERIA CHOOSING BISYNC OR X.25 PROTOCOLS FOR USE WITH DSN/DS

by Carol Hibbard and Mel Brawn
IND Division, Hewlett-Packard Company

ABSTRACT

The new X.25 protocol can be used on the DS-3000 product either for use on a Public Data Network, or on customer supplied communications links. Differences exist between the standard DS-3000 implemented on the Bisync Protocol, and DS-3000 utilizing the newer X.25 Protocol.

This paper discusses the differences in the way data is transmitted on the line, the way internal queuing and buffering is done, and the resultant affect on performance for single and multiple users on the line. The effect of network topology on the decision, and the utilization of computer system resources is also discussed.

INTRODUCTION

HP DSN/DS is a Distributed Systems Network. The user interface provides many services including Remote Commands, Virtual Terminals, File Transfer, Program to Program communications, Remote File Access, and RFA to Message files. The high level services are the same for DS whether using BISYNC protocol or the X.25 protocol. The substitution of X.25 at the ISO levels 1,2 and 3 provides the user with opportunities to trade off improved throughput against somewhat higher CPU load, as well as utilizing private or public X.25 data networks. This paper will deal with the performance differences that characterize the BISYNC and X.25 protocols in a DS/3000 implementation. Cost comparisons will be made. It is assumed that the reader has the companion paper by Brawn, Mel/Hewlett Packard "Performance Characteristics of HP/DSN (DS-3000)". Information and discussion from that paper will not be duplicated here.

DATA FLOW FOR THE X.25 ENVIRONMENT

There are significant changes in the data flow for the X.25 inplementation. The fact that the X.25 protocol is packet oriented introduced changes in the structure. Refer to Figure One. Now as users initiate DS related activities the IOQ entries are prepared at the IODS device. But, when DSIOMX services these IOQ's in response

to the DSMONX activity it passes all active users to DSMONX. DSMONX controls a queue of active users.

The data flow from the user is also controlled by DSMONX. Instead of the request actually transferring data to DSMONX pointers are passed. DSMONX is then able to obtain the next chunk of data, buffer it, prepare for transmission, and send it as required. DSMONX uses an Extra Data Segment for the transmission buffers. This XDATASEG contains enough room for buffering so that the buffer area corresponds to the size of the configured packet size, and enough buffers are provided to correspond to the configured level 2 window size.

For example, with a configured packet size of 1024 bytes, and window size of 7 DSMONX must supply 7 transmit and 7 receive buffers, each being 1024 bytes in length.

Let us assume we are in steady state operation with 3 active users. The level 3 acknowledgement has just verified the receipt of two packets thus freeing up two buffers. DSMONX then takes the next user in the queue and grabs enough data from the users stack or extra data segment in accordance with the pointers, to fill a packet. It buffers this in the next available transmit buffer. It prepares this buffer with the necessary level 3 packet header including address and acknowledgements. It then transmits this buffer to the low level drivers. The user is then requeued to the rear of the dynamic queue. With another available transmit buffer DSMONX services the next user in the queue and transmits the data. This user is requeued. As acknowledgement frames are received and other transmit buffers are released the users are serviced in rotation. Users whose data buffer is completed are dequeued, and new users just beginning activity are added to the queue.

This mechanism reduces system overhead by servicing all potential users at the DSIOM level at every opportunity. Data is moved as required in small chunks. Short interactive user activity is processed expeditiously on a packet by packet basis through this rotational queue.

The Level 3 window allows the control of data flowing between the level three facilities on the systems. It provides for addressing between the various virtual circuits, and the acknowledgement for level three end to end packets.

To complete the discussion, the lower level driver handles the addition of the level two header and trailer. This level one and two activity is responsible for the safe delivery of each packet in the proper sequence. In cases in which retransmission is required the level two requests a retransmission from the transmission buffer controlled by DSMONX.

The completed messages called frames are transmitted by the INP across the communications link to the other computer. The INP contains two buffers so that it can simultaneously transmit and

receive frames. This full duplex line with simultaneous transmission accounts for a significant improvement in performance (compared with the non-simultaneous mode used for BISYNC operation). An additional feature of X.25 that also provides a significant advantage over BISYNC is the multiple outstanding packets and frames which results in a pipe-line effect. The network defines the window sizes. This specifies the number of outstanding packets and frames that can be serviced in the transmission pipeline prior to receving an acknowledgement. With a proper selection of window sizes hopefully continuous transmission can take place without having to wait for the acknowledgements.

The high level DS protocol is unchanged. Thus, for each user request there is a user reply. The programmatic activity must await the return of this reply before it can continue.

THE USE OF THE X.25 PROTOCOL ON A LEASED LINE

The X.25 protocol can be used for DS activity over a leased line. In this case the features of X.25 are implemented, such as: data is packetized, the packet size may vary from 32 to 1024 bytes, the window size for outstanding packets and frames can vary from 1 to 7, and level two and level three still guarantees valid data delivery in sequence. All of the normal DS services are still provided when using this communications link.

Figure Two provides performance information for RFA transfers to a remote message file. The data is non-compressed. The packet size was 1024 bytes, with a level 2 window size of 7. Now there is no perceptable saw-tooth effect. The DS subsystem breaks the user data into packet-sized chunks and keeps the pipeline full. There is no break in performance right up to the maximum user size. This data was taken on system 44's with no other load. The throughput approaches the maximum instantaneous line speed, except for the overhead of the DS Fixed Header and Appendage, and the level 3 and level 2 headers. The time required for the round trip for the DS level reply keeps a single user from fully utilizing the line capacity.

Data is plotted for 19.2 kbit/sec and 9600 bit/sec transmission as well. A user would typically use 56 kbps for a hardwired connection, and 9600 bit/sec for remote use over a full duplex modem. Use of the X.25 protocol is not appropriate over a half duplex line. It requires full duplex service.

The X.25 protocol results in higher CPU utilization. The smaller effective packet size also results in higher CPU load. The CPU load on the local end requires 10.7% for an 8192 user buffer, up to 20.1% for an 80 byte user buffer. At a user buffer size of 500 bytes the X.25 CPU load is 14.4%. This corresponds to 9.4% for the DS BISYNC protocol case.

When there are two simultaneous users utilizing the line the total throughput increases quite substantially. A single user at 500 byte

user buffer achieves a throughput of 2544 char/sec, whereas the aggregate throughput of two simultaneous users increases to 4783 char/sec. This is an increase of 88%. The CPU load on the series 44 also increases to a value of 34.2%. Thus, it is possible to achieve significantly better throughput with multiple users on the X.25 protocol. It is obviously better if the simultaneous use of the line is about equal in both directions. This makes very good use of the full duplex channel with simultaneous traffic. The important elements are making use of the simultaneous transmit and receive, and the servicing of other users while awaiting the return of the user level reply.

Figure Three shows the time per record required for the RFA activity to the remote message file. You will find that the minimum round trip time for a small record is about 110 milliseconds. This corresponds very closely with the value found when using the DS BISYNC protocol.

The effect of system loading will affect the performance. The source of the data may involve disc reads or programmatic manipulation. Other system activity will tend to reduce the throughput. For moderate to heavy CPU loads a reduction of 40 to 60% in throughput would be reasonable. Factors such as disc contention, Image buffering, and relative priorities may greatly influence the observed throughput.

Figure Four shows the throughput for DSCOPY used over the X.25 line. This curve follows very closely the performance experienced in the DS BISYNC protocol. The line utilization for a single user is not particularily high. This is because of the time required for disc accesses and buffer packing. You will notice a significant effect based on the blocking factor, almost a three to one improvement can be achieved with optimal blocking, at high line speeds.

The CPU utilization ranges from about 7% to 8% with various blocking factors with this non-compression example. The disc accesses range from slightly over 20 to below 1 per second with a blocking factor of 16. When compression is used with a Fortran source listing containing 60% redundancy the apparent throughput increases to 8900 characters with a blocking factor of 16. The CPU load increases to about 16.2%.

When multiple users share the line the aggregate throughput reaches 14,700 char/sec (compressing with the Fortran source file). The CPU load increases to 23.8%.

The characteristics of the X.25 protocol can be utilized on a leased line facility to provide greater throughput, but at a cost of somewhat higher CPU load, compared with DS using BISYNC protocol.

X.25 ON A PUBLIC DATA NETWORK

X.25 Public Data Networks (PDN) are available for distributed systems. The PDN provides a network designed to support

communications activities between many users that are geographically distributed. The principle is to take advantage of the economy of scale. By sharing high capacity communications links between many users the cost per user is reduced. The PDN provides many services. These services include establishing and routing the virtual circuits between various destinations, provides a standard electrical interface between the user and the network, delivery of packets in sequence and free from data error, buffering and queueing as users share the communications facilitites, etc.

The PDN normally handles the lower three levels of the ISO model. The first is the Physical level. This includes the electrical connection specifications for voltage levels, polarity and impedance. The second level is the Data Link level. This level provides the point to point integrity. It provides for error free transmssion of frames, and handles acknowledgements and flow control. The third level is the Network level. It handles the host to IMP interface, the subnets, packet handling, routing and channel allocation, congestion management, and the virtual circuit mechanism or datagrams.

The higher levels of the ISO model provide additional features often handled by the computer system or application. Level 4 is the Transport level. It includes host to host communications, multiplexing, directory, flow contol, and synchronization. The 5th level is Session level. It includes the user interface to the network, addressing, binding or connection establishment, session management, and process to process communications. The 6th is the Presentation level. It includes text compression, encrytion, virtual terminal and file transfer. The 7th level is the Application level. It includes services such as network transparency, problem partitioning, distributed data bases, computation, and operating systems.

In HP DSN/DS the higher levels are included in the DS software implementation. All of the normal DS capability is still available when used over the X.25 PDN. These include the Virtual Terminal, Remote Commands, Remote File Access, File Transfer, Remote Data Base Access, Program to Program communications, and RFA to message files. Thus, all normal DS activities are still available, plus an additional capability has been added. This includes the support of asynchronous terminals over a PAD. The PAD is a Packet Assembler Disassembler. It provides a direct interface for support of asynchronous terminals on the X.25 PDN.

The PAD receives the transmissions from the terminal character by character. When the packet is full, or when an input terminator is received the PAD completes the packet and sends it to the host system. Upon receiving a packet destined for the terminal the PAD handles the handshaking betweeen the terminal for the display and transmission of the outgoing data and buffering of incoming data.

This capability provides a mechanism so that terminals which are scattered geographically can utilize the X.25 network for connection to the host computer. In the normal mode of operation the terminal site has an asynchronous modem which provides access to the PDN via

the PAD. The PDN routes the traffic to the host. Thus, a session is established on the HP 3000 on behalf of this terminal user.

A computer network might consist of five computers and 20 terminals all geographically dispursed. Using standard leased line or dial up line communications the determination of routing and line speeds might be a difficult problem. Unless the network of computers were fully connected the user routing would have to be considered for each application or activity. It might also be difficult to provide an easy link between each terminal and computer system.

The network services of the X.25 PDN provides for the establishment of direct connections between each computer and terminal. Each of the computers would be connected to the PDN with an appropriate communications link. This would normally be a 4800 or 9600 bit/second leased line. Higher and lower speed are also available from some networks. The terminals would achieve access to the PDN through dial-up asynchronous lines. Users on any computer can achieve a virtual circuit and perform any standard DS activity to any of the computers in the network. Thus activity over a single line to the PDN can handle various simultaneous users between any of the computers in the network. Similiarly, the terminal users can establish a session on any of the computers in the network. These capabilities may greatly simplify the rational connection between computers and terminals for a wide variety of applications. It may also decrease the cost for the network by providing the required inter-connectivity with fewer lines. However, the cost structure between normal leased or dial up lines and a PDN varies greatly from country to country. There may be substantial differences in relative costs for a given network depending on the specific tariffs.

The PDN can provide significant improvements in the reliability of the connection. In networks utilizing leased lines or dial up capabilities the cost of multi-connectedness may be very high. This results in networks that are susceptible to error conditions on single lines or nodes. The PDN normally provides multiple paths through the network. Traffic can be rerouted in cases of transmission difficulties. In our experience the transfer of files internationally normally works better through the appropriate X.25 PDN's with gateways between the networks than with dial up services.

EFFECT OF THE PDN STRUCTURE ON PERFORMANCE

The PDN network is made up of a number of computer systems connected by a number of communications lines. The computers are referred to as IMP's in the Arpanet terminology. IMP stands for Interface Message Processor. The IMP handles all of the activities corresponding to levels one, two, and three. They establish the virtual circuits, handle routing, buffering, queuing, etc. Depending on the relative locations of two user computer systems the traffic between them might be serviced by a single IMP. If so the delays and queuing through the PDN would normally be reduced. If the virtual circuit must traverse two or more IMPs there is potential for a far

greater variation in performance.

The X.25 PDN provides for the packetizing of data. In most networks the default packet size is 128 bytes. The network provides for pipelining with a typical level 2 window size of 7. This means that at level two seven frames can be outstanding at a time. The next frame must await acknowledgement of one or more of the previous frames. At the higher level the DS request and reply structure still exists.

The level 3 window provides for flow control for each individual users. In DSN/DS each user is queued at the DSMONX level. On many PDN networks the level 3 window size is 2. This allows each user to have up to 2 outstanding packets. At that time an acknowledgement must be received in order to send additional packets for this user. The level 3 acknowledgement is normally handled DCE-to -DTE. If the network only supports END-to-END level 3 acknowledgements significant performance degradation would result.

A user request must complete before the program may continue. This means the packets corresponding to the user data are sent sequentially to the remote end. Then the request is serviced on the remote computer and the DS user reply is prepared and returned to the originating end. The time required for this round trip through the network has a profound effect on the throughput.

PERFORMANCE THROUGH AN X.25 PDN

The test set up included two Series 44s. They were connected through the Telenet X.25 network. A 4800 bit/second full duplex line was provided to the local Cupertino California, (San Francisco bay area) port, and a 9600 full duplex line to the Denver, Colorado port. This configuration guaranteed that the traffic would have to traverse the X.25 network, and not merely be serviced by a single IMP.

Tests of RFA to a message file were made on various user buffer sizes. Tests of minimal user record size of 2 bytes required an average of 844 milliseconds for the elapsed time from request to reply. The actual figure ranged from 536 msec up to 3.032 seconds. Out of 60 samples 10% exceeded one second. These tests were conducted at various times, but in the 8 to 10 PM time frame. I believe the same activity during peak traffic loads on the network would result in some performance degradation.

Tests of RFA with a record size of 500 bytes produced an average round trip for the request - reply of 2.849 seconds. The results from 100 samples ranged from 2.352 to 12.489 seconds. This provided an effective throughput of 176 characters per second. With user buffer sizes of 2000 bytes effective throughput of 249 char/second was achieved. Obviously the transmission of compressible data would improve these figures.

Tests of DSCOPY with no compression produced results of 201 char/sec through the network. With compression, using a Fortran source file

with 60% redundancy the effective throughput reached 486 char/sec.

FACTORS AFFECTING PERFORMANCE

There are many complex factors which affect throughput in a network which utilizes an X.25 PDN. The affects of the PDN itself are highly variable and essentially uncontrollable. The instantaneous network traffic is unpredictable. There may exist heavy steady network load. There may be bottle necks on certain communications links or IMPs, which result in large queues.

On the local or remote user system the normal CPU load, priority levels, Image queuing, disc access limits and other performance related factors still pertain. A further limit exists compared with the DS BISYNC communications network. In the BISYNC network each communications link is serviced by a separate INP. The traffic through a given link may be easier to characterize. The speed of the lines may be considered in conjunction with the anticipated traffic loads. On the X.25 PDN a single INP normally can provide the link to the network, although multiple lines can be used. The traffic between this computer and all other computers and pad terminals share this single line. The simultaneous activity between nodes and PAD terminals may cause peak traffic congestion problems.

The performance can be optimized in those cases in which the amount of data per request is increased, thus for a given application the total number of request-reply pairs is decreased. For example with a network round trip delay of 2 seconds a series of 80 byte user records might require 2.2 seconds per record, whereas, 10 records serviced with a single user write might require only 2.8 to 3.5 seconds. This offers a significant improvement in overall user performance without requiring changes in the characteristics of the X.25 PDN.

Psychological concerns must be addressed. Activities requiring interactive access usually make the delays through the network very apparant to the user. Depending of the nature of the application the delays may seem troublesome. Applications requiring program to program activities or file transfers may not be so susceptible to the apparent network delays. Once the data returns, and begins to be displayed, the feeling of performance seems satisfactory to most users, i.e., the display of the data is not as bursty as in some other subsystems.

CHARACTERISTICS OF TERMINAL SUPPORT OVER THE X.25 PDN

The support of asynchronous terminals over the X.25 PDN may offer significant cost savings and ease of connectability. Depending on the nature of the activity the cost of individual lines between terminals and computers may be high. PAD support offers advantages where terminals are dispursed geographically and where the terminals need access to various computer nodes in the network. The X.25 network establishes the virtual circuit directly to the destination

node computer. There is never any need to operate the terminal through intermediate computer nodes. The terminal can also easily change its destination computer without having to redial the PAD.

The topics of flow control between the terminal and computer must be addressed. Some applications use fairly large amounts of data per exchange. In applications that require heavy interaction between the computer and terminal for many single character or short exchanges the delays in the network may profoundly affect the performance characteristics of the application.

Our experience indicates that for terminal support over PADs the turn around delays through the network may range from 600 or 800 milliseconds up to 6 or 8 seconds. One to three seconds has been typical. During busy daytime hours 4 to 6, or even 8 seconds has not been unusual. The delays a user might experience depend on many factors such as network loading, network routing, congestion and flow control. It is difficult to anticipate performance characteristics of the PDN. Many users feel that the PAD support over the PDN is not much different from an asynchronous terminal supported over a 1200 baud modem.

COST COMPARISONS USING X.25 PDN

Charges for service on the X.25 PDN may consist of four elements. There is the initial installation charge. Then there is typically a service change for the account. There is a monthly charge for each computer connection to the network, and this depends on line speed. There is sometimes an hourly connect charge. There is normally a volume charge on a per packet or a per frame basis. Finally, when inter-network activity is being utilized, there may be additional charges for connect time and transaction volume. In some tariffs there are charges for a larger number of simultaneous virtual circuits, i.e., the maximum number that may be required at a time.

Charges for terminal support over the PAD typically involves an hourly connect charge plus the transaction volume charges. Of course there is the additional charge for the computer to network connection.

These examples use costing data supplied by Telenet, and the Bell Telephone Company in the USA; and for the DATAPAC service and Dataphone Digital Service in Canada. There is an extremely wide variation in pricing policies in various countries and from various vendors. The prices for X.25 services seem high compared with the price of leased lines in the USA. In many other countries the relative price of leased lines is significantly higher, so X.25 seems better by comparison. A determination based on current costs for each situation will be required.

Refer to Figure Five. Consider 10 computer nodes located in various places with an average link distance of 17 miles. The minimum connection using private lines requires 9 links. In the US this might cost:

9 (lines) * $120 per line = $1080 plus 18 (modems) * $225 (per month) = $4050.

This total cost is $5130 per month, and provides 9600 baud connections between computers. In general the configuration would not provide full interconnectivity, so from the applications standpoint the user might have to utilize intermediate nodes.

The same network configuration using an X.25 PDN (using Telenet charges) would cost:

10 (lines) * $925 (per line for 4800 bit/sec) = $9,250 per month; or:
10 (lines) * $1400 (per line for 9600 bit/sec) = $14,000 per month.

The packet charges are $1.55 for 128 byte packets. For a heavily utilized line at 4800 bit/sec the packet charges could approach $25 per hour.

With the X.25 PDN each computer can access any other node at any time so the user operations will be easier than through intermediate nodes.

Using charges from Canada for the leased line facilities assuming an average distance of 17 miles we would have:

9 links at	$ 600 /month	$ 5,400	
18 modems at	$ — /month	$ —	
	Total	$ 5,400	/month.

This data is based on 4800 bit/second, full duplex service.

To provide the same service using DATAPAC the charges are as follows:

9 computer links to the network at 4800 bit/sec $270 + (19 * $3) = $2,943 /month.

There is one virtual circuit per link. Additional potential virtual circuits cost $3.00 per month. The Packet charge is $0.19 per 1000 packets.

Example Two. If we consider the same configuration, except that now the average link distances are 1300 miles. The line costs for the US case providing 9 leased lines is now $11,700 /month. The total monthly cost is $15,750, including modems. The price of the same network over X.25 PDN is unchanged from above. The total network charge for X.25 will depend on packet charges. This total price may be either lower or higher than the private lines, depending on traffic charges. Then the overriding consideration would be convenience, throughput, connectibility, etc.

Refer to Figure Six. Consider a fully connected network consisting of four computers located at Seattle, San Diego, Boston, and Miami.

To achieve a fully connected configuration it would require the following leased lines: 2 at 1500 miles, 2 at 2500 and 2 at 2800. The line cost would be $9,900 per month and the modems $2800 for a total monthly cost of $12,700. The X.25 PDN would provide a fully connected network with only 4 lines. Four lines at 9600 would cost $ 5,600 per month. However, packet charges could run up to $40 to $50 per hour on a fully utilized 9600 bit/second link. The determination of whether to choose standard leased lines or the X.25 PDN depends on factors such as connectibility, line utilization, performance considerations, and cost.

Consider Figure Seven. This show a network consisting of Montreal, Halifax, Toronto, and Vancouver. The mileage figures are shown. The rates for using the 9600 bit/second Dataphone Digital Service 24 hours per day are given below:

Montreal - Toronto	$2035 /mo
Montreal - Vancouver	4281
Montreal - Halifax	2846
Toronto - Vancouver	4211
Toronto - Halifax	3412
Vancouver - Halifax	3731
Total	$21,203 / month

The service utilizing the DATAPAC X.25 PDN follows:

4 Links at $390 + (19 * $3) = $1788 / month (including 20 switched virtual circuits per connection).

The charges for packet follows:

Montreal - Toronto	$0.50 / 1000 packet
Montreal - Vancouver	1.30
Montreal - Halifax	.60
Toronto - Vancouver	1.24
Toronto - Halifax	.78
Vancouver - Halifax	1.44

If we assumed one full hours traffic (i.e., 100% utilized) per day on each line the packet charges would $210 per day, or $4620 for a 22 day month. This is still quite reasonable compared with the DDS service. Obviously the amount of traffic must be carefully considered.

PAD EXAMPLE

For another example consider a situation in which terminals in 100 different locations require access to the computer for an hour per day, 20 days per month. For public dial-up lines in the US, assuming 35 cents/hour phone expense the monthly bill would be $42,000.

100 (hour/day) * 60 (min/hour) * .35 ($/min) * 20 (days/month) = $42,000 per month.

This includes only operating expense, and does not include the

purchase price of the modems.

For service on an X.25 PDN there are connect charges and packet charges, plus the link to the computer. For Telenet billing:

100 (hour/day) * 3.90 ($/hour) * 20 (days/month) = $7800

The number of packets might vary from 3000 to 20000 per hour depending on the nature of the application. At 5000 per hour the packet charges might be:

100 (hour/day) * 20 (days/month) * 5000 (packet/hour) * 1.55/1000 ($/packet) = $15,500.

The line cost to the computer would be $1400 per month at 9600 bit/second. Thus, the total cost using the X.25 PDN might be $24,700 per month, about 60% of the dial-up line approach.

For the corresponding Canadian charges the data included in the examples above is still valid, however, there are two additional charges. There is a connect charge of $0.025 per minute ($1.50 per hour). There are also charges of $0.35 per 1000 packets in addition to the packet charges mentioned above for the computer link.

From the applications stand point the throughput characteristics of the network, compared with a dial up approach, might be an important factor. In general, the short interactive types of activities might be impacted due to the turn around delays on the PDN. The packet charges could vary widely. It might be necessary to provide more than one line to the computer to handle 100 terminals.

SUMMARY

The availability of the X.25 protocol for HP DSN/DS allows one to trade off performance and cost with CPU overhead. For leased lines between the computer nodes the X.25 protocol can provide better throughput for multiple users, for transfers that exceed the configured packet size, and for short interactive activities in the presence of other simultaneous users. The trade off includes somewhat higher CPU load. The Line should be configured with a packet size of 1024 bytes, a level 2 window size of 7, and a level 3 window size of 7.

For network activities over the X.25 PDN improved connectibility and different cost factors can be achieved, but with possible significant differences in the round trip delays through the network. The support of asynchronous terminals geographically dispersed is also a beneficial capability of the PDN. The PDN will specify the packet size and allowable window sizes, with packets of 128 bytes, and level 3 window size of 2 being common values. Although the CPU load to handle these smaller packets is increased, the fact that this load is paced by the speed of the line generally keeps the impact on the system from being a significant problem.

Since terminals on the PAD are normally serviced at 1200 bits/second the 9600 bit/second line between the network and the host computer might easily service 8 to 12 terminals operating at full speed.

FIGURE ONE
DATA FLOW (X.25)

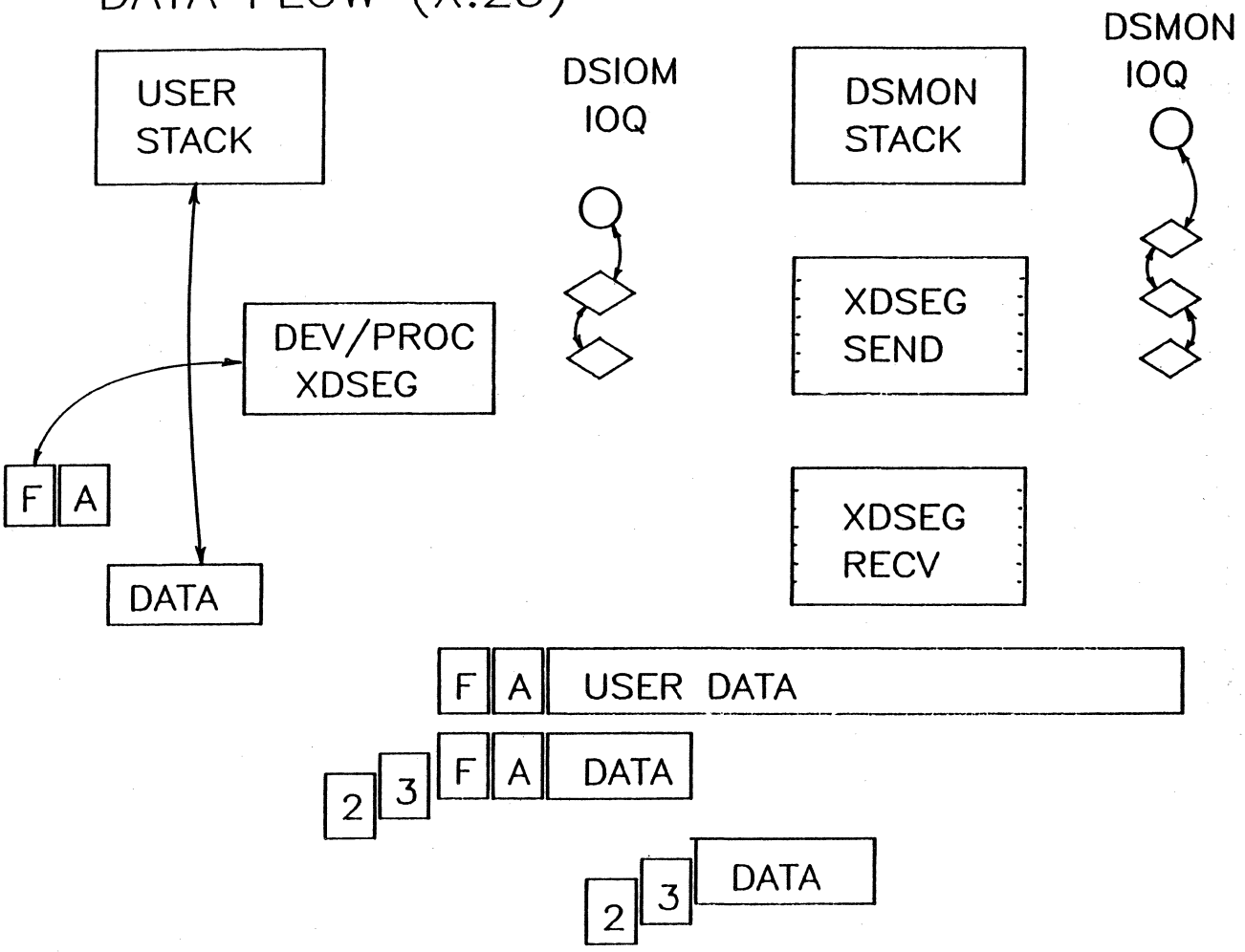


FIGURE 2, RFA PERFORMANCE (X.25)

MODEL 44's, No Other Load

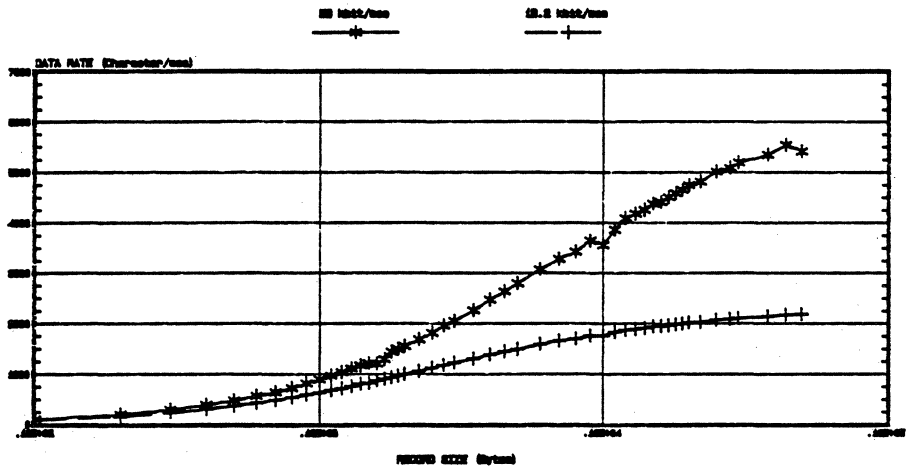


FIGURE 3, RFA TIME/RECORD (X.25)

MODEL 44's, No Other Load

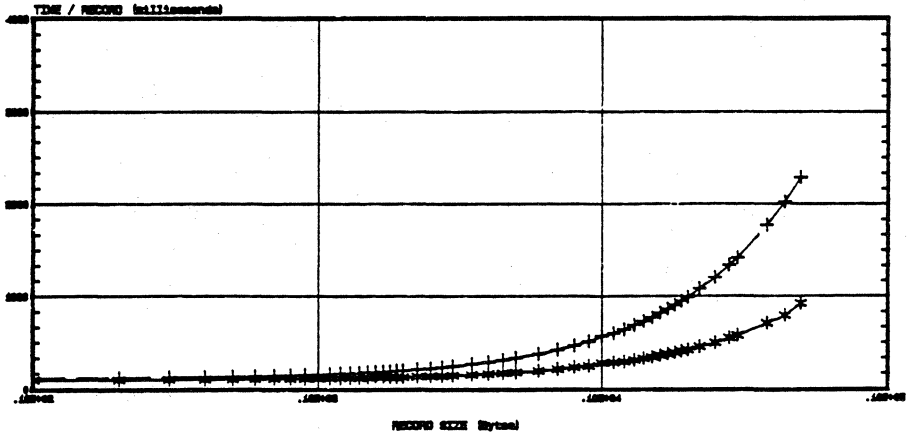


FIGURE 4, DSCOPY PERFORMANCE (X.25)

MODEL 44's, no other CPU load

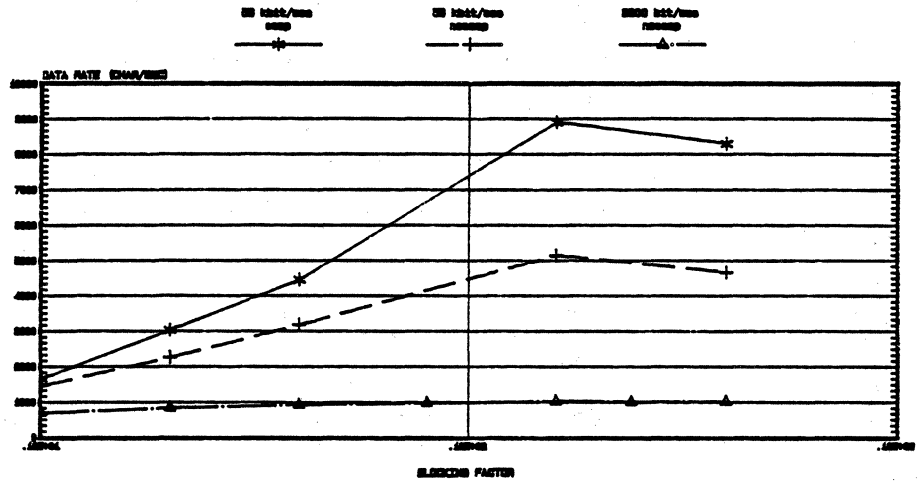
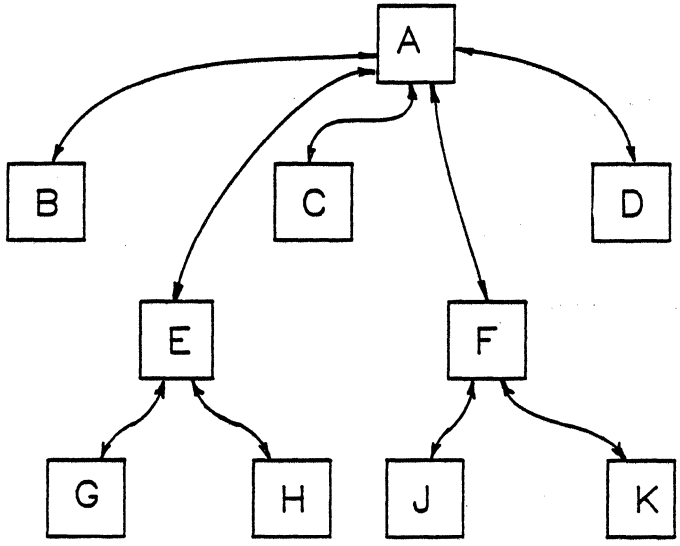
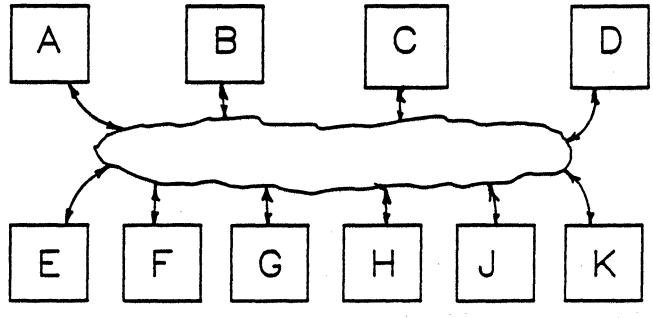


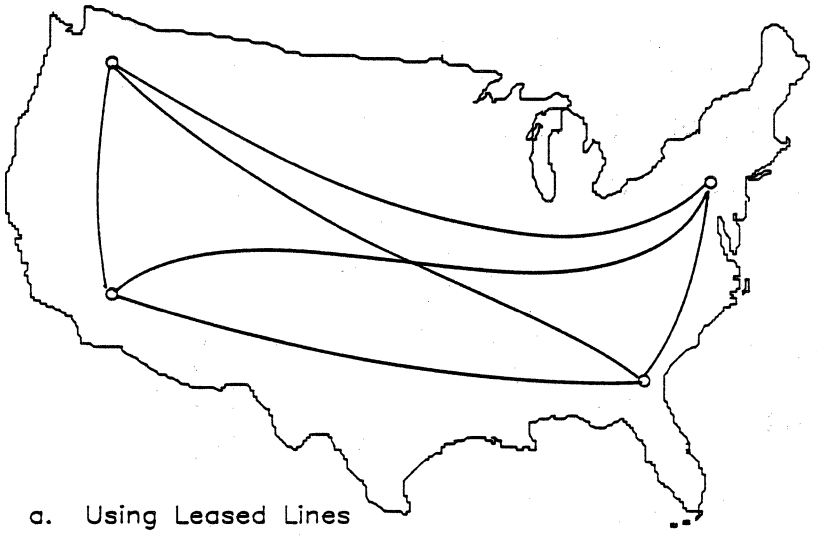
FIGURE FIVE
10 COMPUTER NODES

a) using leased lines

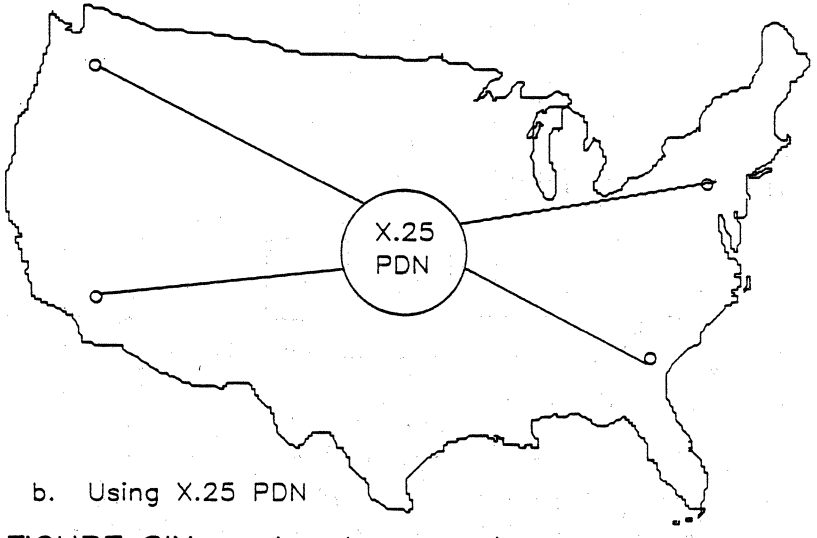


b) using X.25 PDN





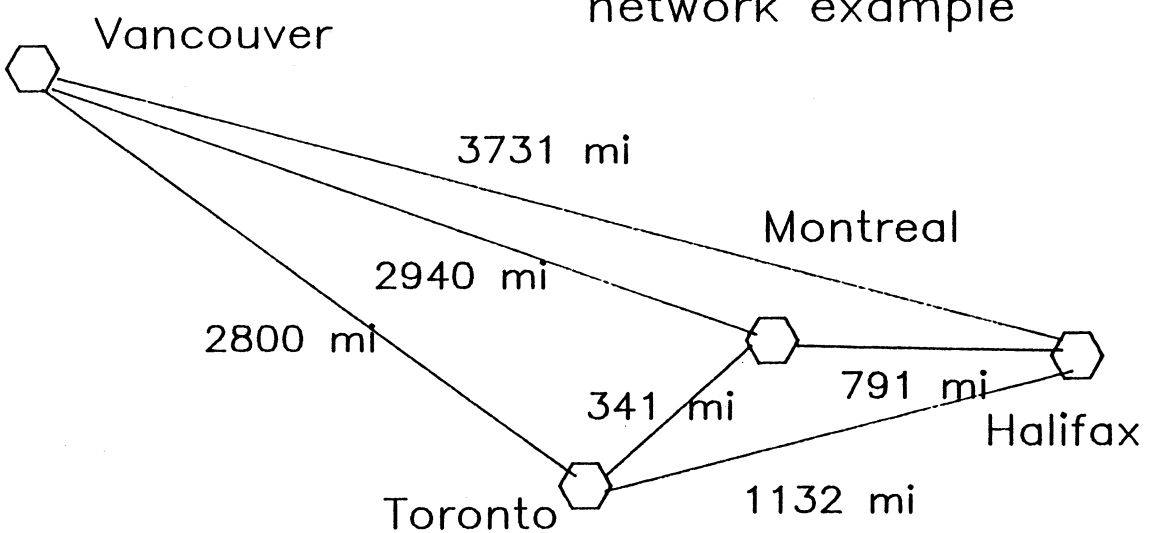
a. Using Leased Lines



b. Using X.25 PDN

FIGURE SIX network example

FIGURE SEVEN
network example



Beyond DSG 3000: Comprehensive Decision Support Systems Using Graphs

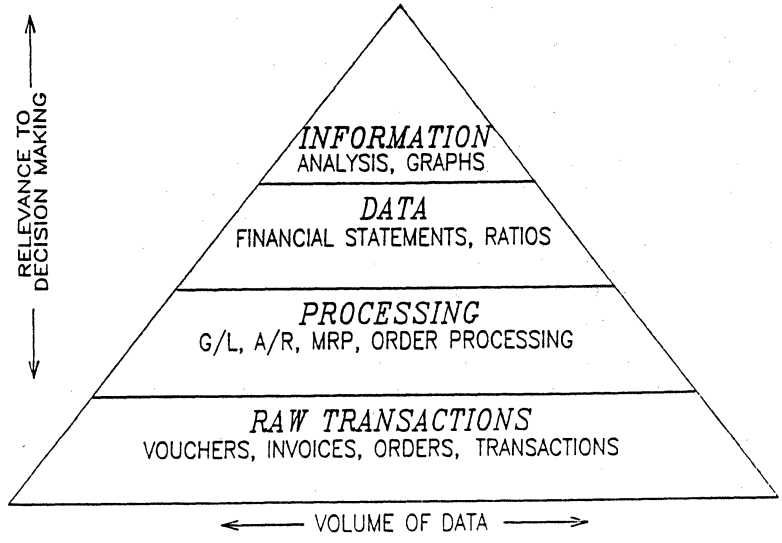
by

Robert F. Hoel, Catherine Y. Fleischmann, and John W. Ellis
Colorado State University

As managers become more sophisticated decision makers, they will need more advanced computerized support systems. We believe that the most useful decision support systems will integrate (1) information analysis, (2) graphical output, and (3) ease of use. This was the type of package the Colorado State University, College of Business was seeking to use on its HP 3000 for purposes of demonstrating and teaching effective business decision making.

An integrated decision support system provides the most relevant decision making information to managers. An organization's information flow can be thought of as a pyramid. The massive volume of the actual individual transactions make up the base of the pyramid.

Chart I
An Organization's Information Flow



These items are processed by many different systems to create various types of data that are more relevant to the manager. At the top of the pyramid the volume of data needs to be reduced. Managers must have usable information and must have procedures for analysis. While it is important for managers to be able to dip into the lower parts of the pyramid occasionally, they must not be overwhelmed by too much raw data. The most useful management-oriented data is at the upper levels of the pyramid. From the upper pyramid came key decisions that affect the operating success of an organization, and it is here that decision makers gain their most important insights into the performance of their organization. Insight is generated by managers as they interactively survey and analyze their data.

Graphics are invaluable as the primary output because of the ease with which a user can assimilate large quantities of data in pictoral form. A recent Harvard Business Review article[1] described the appeal of graphics:

> ...Computer graphics offers two basic benefits to all managers. First, it saves on one of the most coveted resources available to managers--time. Such savings occur in the time required to a) interpret the data, b) communicate a complex set of findings to others, and c) supervise the production of a final report. Because the time used in processing data is essentially nonproductive, these savings free managers to engage in other, more productive functions.
>
> Second, computer graphics helps managers better perform one of their most important functions--decision making. This happens because a) visual information can be digested more readily, so that managers would, in a given amount of time, be able to acquire a larger pool of relevant information; b) trends or deviations from the norm are easier to depict through graphics, so that managers would have access to a more accurate system of exception reporting or trend analysis; and c) the rapid response of the interactive computer graphics systems enables managers to ask the 'what if' questions without feeling guilty about having to consult others and helps them personally test out several contingency plans in a relatively short period of time."

Graphics serves the dual purpose of a communication device and a decision aid. Graphs are appropriate in boardroom presentations, and many types of decision-oriented work sessions, publications, and many other communication needs. Graphics help analyze past trends and identify turning points. Managers can identify opportunities, project trends, and evaluate "what if" situations.[2]

Shortcomings of Currently Available Systems

Most graphics packages that are currently available for the HP 3000 fall short in many areas. The problem areas include the ease of use, the purpose of the package, available networking, and expandability. We will use the DSG/3000 package for purposes of criticism because it is so widely used.

Ease of Use

The three major types of selection devices with graphics packages are: 1) English-like commands, 2) small programs accessing a series of subroutines, and 3) menu decision trees. The first two types are fine for technical people, but tend to exclude a non-technical manager from interacting with the package. It is difficult for managers to request a series of graphs from data processing when the manager is examining "what if" kinds of situations. The turn-around time here is also a problem when managers wish to see a graph immediately. The menu decision trees in DSG/3000, for example, are easily understood by non-technical managers but are cumbersome to use when changing a graph. DSG requires a user to know what series of menus to go through to change something as simple as a line pattern or switch from a bar to a pie chart. The criticism is not that menu trees are difficult to understand, but they are time consuming.

Purpose of the Package

The packages currently available are primarily for making presentations. They create high quality graphics but perform very few statistical and analytical functions. DSG will allow the user to generate the moving average of a line but not a linear or exponential regression. It is practically impossible to extend a line either by a step function, growth rate, or trend unless the data is calculated by hand. While DSG will perform various arithmetic operations, it will often not provide the user with basic statistical information about these lines. For example, statistics not included in DSG/3000 are range, standard deviation, regression r squared fit, regression slope, and number of valid data points. These statistics provide valuable information to managers and their absence requires valuable time for calculation by hand. Managers are likely to make decisions without these data values before doing the computations themselves.

Networking

Since a major purpose of graphics is communication, several managers may need to see the same graph as well as analyze the same information. To save time and material, the manager should be able to see authorized material created by other managers and

send his own graphs to others. This creates problems with data security and integrity. To see a graph a user created under DSG provides security because the second user must know the data file and chart names to receive the graph. However, DSG does not fully protect data integrity. An unscrupulous second user is able to change data and graphs to meet his own purposes.

Expandable Features

Most of the vendor graphics packages available are not expandable. A user must purchase the entire package whether or not they will be using all the features of that package. If users find they have outgrown the package, they must purchase a new package that may or may not operate in a similar manner. While DSG does offer various add-on features, it currently cannot be expanded to include more advanced analytical functions. On the other hand, DSG may offer too many features for a first time graphics user. This person might have to buy a simple package at first (to justify the cost) and trade up to DSG in the future. The user would have to learn how to operate a new package. This is time consuming and costly.

Ideal Decision Support Features

Problems with the currently available graphics packages make it difficult for companies to consider graphics as a true decision support system and genuine aid for non-technical managers. An ideal system should allow managers to apply their analytical skills without being dependent on data processing personnel. Managers also need an interactive format, analysis capabilities, text and tabular features, potential for expansion, user-specified default parameters, networking, and multiple output devices.

Interactive

The ideal package should allow for on-line user interface from the manager's desk or work station. The manager should be able to operate the package easily without learning any code or torturous "English-like" language. All the functions should be evokable at any time--this means menu trees should not be included. This evokability could best be accomplished through the use of a dedicated keyboard. Each key would have at least one special function that when pressed may produce the necessary form to be filled in according to the user's needs. The keys would be labeled so that a manager would not need to memorize key functions. Changing the features of a graph (i.e. bar to pie) would mean pressing one or two keys.

Analysis Capabilities

The desired package would provide managers with a wide range of analytical tools including regression, moving average, inter-line and inter-point calculations, confidence limits, statistical values, and a technique (similar to operating an HP hand-held calculator) capable of solving and storing user specified algorithms. The user should be able to perform these analyses on 1) any data that is stored on the HP 3000 or 2) data that the user directly enters himself. This enables the manager to browse through data to easily discover trends, turning points, and opportunities. An example of this type of analysis process is shown in Graph I - Graph IV at the end of this paper.

Text and Tabular Capabilities

The desired package would provide high-quality presentation materials. Also, a user must be able to generate slides of text and/or tables to tie together presentations. Again these should be easy to produce by pressing only a few keys. The text should allow for various letter sizes and font types as well as repositioning and rotating of words. In producing the tables, the user could incorporate mathematical and statistical functions. The user could provide actual figures as well as graphs to his audience.

Potential for Expansion

A package that contains all the previous specifications would overwhelm many managers. Not all features would be used immediately--decision makers learn how to use a few sophisticated techniques at first and later learn to use more advanced techniques. Therefore, users should be able to start with a simple package and add on the various features as they become useful. An ideal system would allow users to learn one basic package and simply add new functions to the same package when needed.

Specified Default Parameters

In addition, the user should be able to easily specifiy and modify default parameters for 1) the total fiscal period, 2) numer of time periods per graph, and 3) graphical features (i.e. axis, tick marks, legend size, frame, etc.). The user could then quickly point at a data set and hit a graph key which would produce a graph according to his default parameters. If the graph needs minor changes, the pressing of a few keys would enable the user to see the new graphs quickly without having to specify all parameters.

Networking

A graphics decision support system should allow users to interchange graphs without jeopardizing security. This could be done by providing each manager with his own user name and the ability to send a graph to other managers. The managers would be able to see the graphs and use any analytical functions to change the graph, but would not be able to change the actual data or the original user's graph. This swapping of graphs could be done on-line by the managers and would not involve the data processing department.

Saving time and money is a critical feature of networking. Managers would spend less time generating hard-copy output for other managers. Each manager could see the graph on his own screen, thus reducing the need for paper and other costly supplies.

Also, communication between various managers in different departments, different locations, and different organizational levels would increase due to constant sharing of information.

Output Devices

A manager should be able to easily generate output on a plotter, printer, and terminal by pressing a specific output function key. The high resolution output would be suitable for any presentation or publication needs.

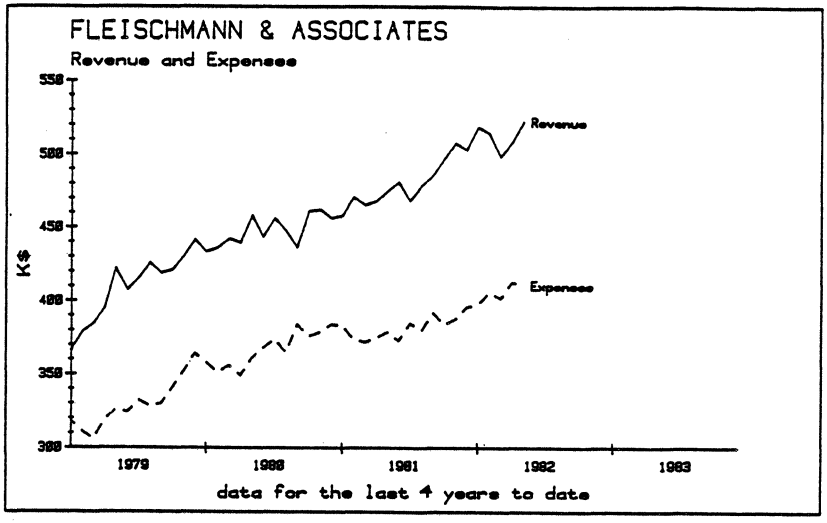
The Decision--Support System at CSU

We are only aware of one business graphics package that now has most desired characteristics of an ideal graphics decision support system for the HP 3000. The package is ANALYST/3000 by Interactive Solutions Corporation in Fort Collins, Colorado. It contains 30,000 lines of code of which 8% is devoted to graphics and 92% to data handling, user interface, and analytical functions. ANALYST has several features and ranges in price from $8,000 to $25,000. Features can be added at any time to the basic package. The College of Business has decided to use the ANALYST/3000 to teach future managers how to interpret and analyze their data to make more efficient and timely decisions.

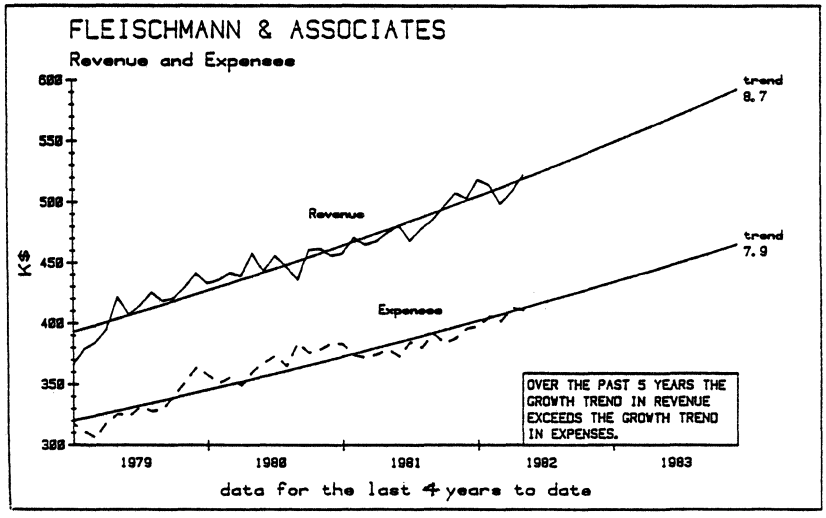
As educational institutions incorporate sophisticated decision support systems in the classroom, students will be entering business with a wider range of decision making abilities. These new managers will need advanced decision support systems available or their unique abilities will be wasted. It is the responsibility of data processing departments to provide these new users with sophisticated packages that are

non-technical and easy to operate. One of the most important tools that should be available is a easy-to-use decision support system using extensive graphics and a wide range of analytic aids.

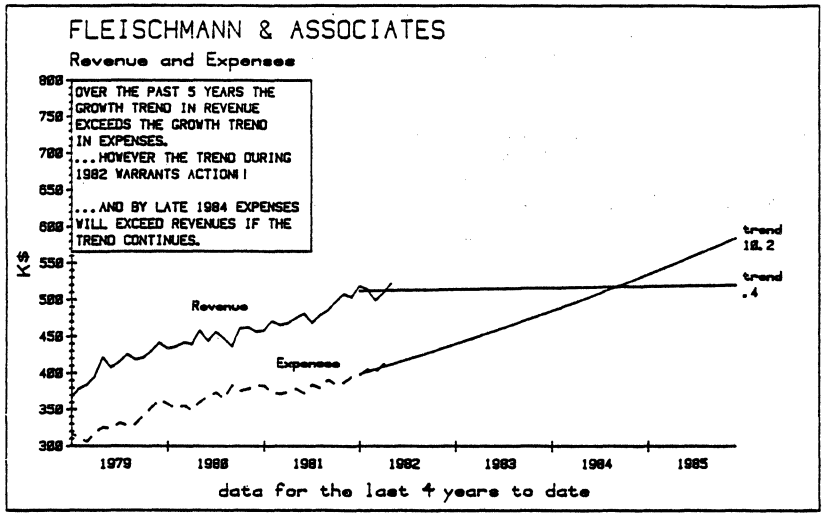
Graph I



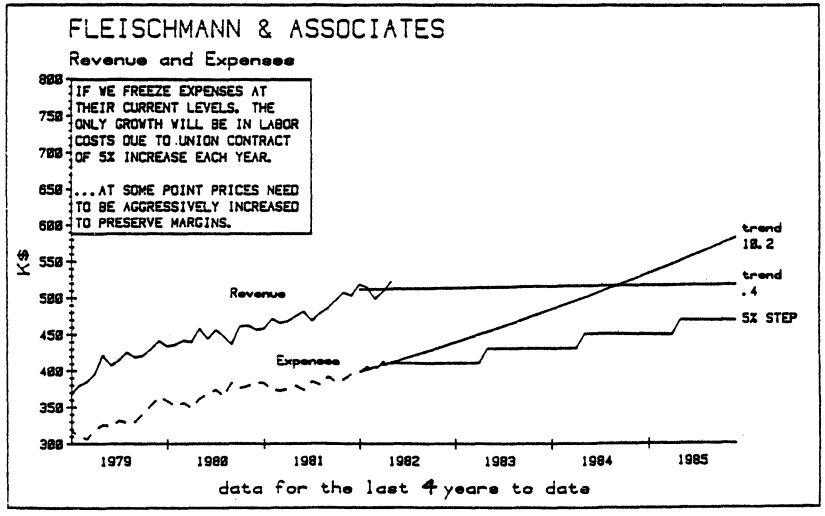
Graph II



Graph III



Graph IV



Footnotes

[1]H. Takeuchi and A. H. Schmidt, "New Promise of Computer Graphics," Harvard Business Review, January/February, 1980.

[2]For further discussion of this general topic, see Malcolm L. Stiefel, "Software Revenues to Parallel Graphics Boom," Mini-Micro Systems, July, 1982, and David Friend, "Graphics For Managers: The Distributed Approach," Datamation, July, 1982.

The Block-Mode Conspiracy--A Case of Ignoring User Input

by John Hulme
Applied Cybernetics, Inc.

In the early 1970's, HP was faced with a decision which has since affected the lives of many thousands of HP3000 users. In those days HP didn't sell application packages; they sold top-rate equipment and just enough software to make it useable. When you bought an HP3000 computer, you got an operating system (MPE), a couple of compilers, a few utilities, and maybe a database facility (IMAGE/QUERY). With these powerful tools you could develop and execute any kind of program you might need.

The only thing lacking was a good, user-oriented data entry facility. Commercial users wanted formatted screens, but there was no easy way to implement them with existing languages. In the absence of such a facility, HP may well have been losing hundreds of orders a year, and customers who did buy an HP3000 often decided to get terminals from some other vendor rather than pay $4000 apiece for HP's expensive 2640's.

The crucial decision was whether HP should provide such a facility, and if so, what kind of facility it should be. If HP didn't do it, one or more independent vendors undoubtedly would, and HP had both the resources to do the job right and the reputation to generate widespread user acceptance of the finished product. It was clear that developing a user-friendly formatted screen facility would not only be a great boon to HP3000 users; it would be in HP's economic self-interest as well.

This new product was destined to be known as VIEW, V/3000, and eventually V/PLUS; but before development could begin there were some tough questions that had to be answered:

1) What was meant by a formatted screen facility?
2) Would specialized terminals be required?
3) Would the facility operate in character-mode as well as block-mode?
4) Would it fit in with the existing operating system without impacting performance?
5) Would the facility be accessible from application programs, or operate as an independent package?
6) Could such a project be justified economically?

One can only speculate as to the conversations that might have taken place in the HP planning sessions with respect to these issues. We can, however, evaluate these questions from the vantage point of the user community.

What Do We Mean by a Formatted Screen?

Before terminals were invented, most data entry was accomplished by punching numbers and letters on 80-column cards. The "card format" was nothing more than a precise description of which fields were located in which columns and how each field should be interpreted.

Batches of these cards were read by a batch input program and checked for consistency. Entries that contained recognizable errors were rejected and identified in a batch exception report while those that were judged valid would be processed and/or stored as individual records on a disk or tape file. As before, each type of record had a "record format" defining the position and content of the various fields contained in the record.

With the advent of timesharing facilities and multi-processing operating systems (such as MPE) it was no longer necessary for a computer to be dedicated to one program at a time; it could be shared by many users concurrently. This eliminated the necessity to process data in batches, and made interactive processing practical for the first time.

Instead of key-punching and verifying a batch of cards, the former key-punch operator could now enter data directly into the computer, through teletype devices at first, and more recently using video display terminals. A "screen format", similar to that used for punch cards could be defined identifying the sequence, location, and meaning of fields typed on the terminal in any given input line.

The obvious advantage of direct entry into the computer is the feature of immediate feedback. Instead of getting a printed report of all the entries that had to be revised and resubmitted, the operator could be notified of invalid data as it was entered, and the error could be corrected immediately.

Terminals are used in one of two basic modes: scroll-mode (line-by-line input), and non-scroll-mode (full-screen input). Note the important differences between the two.

In teletype- or scroll-mode, the user enters data on the "bottom line" with the preceding several replies visible on the lines above. This is the mode generally used for the question/answer type of dialogue in which a user is expected to respond to one prompt at a time (note that each successive question may be determined by replies given to earlier questions).

Scroll-mode is the standard method of communicating with MPE. It is also used for line-by-line input in virtually all of the HP utilities, including QUERY, EDITOR, and BASIC. By treating each line as a fixed-format card-image, this type of input may be viewed as a rudimentary version of the "formatted" screen.

The true formatted screen, by most definitions, operates in non-scroll mode. The screen format is analogous to a preprinted form with several blank areas waiting to be filled in. The "background" portion of the form typically contains headings, instructions, and descriptions of the various data fields (name, address, part number, quantity, etc.). The "foreground" portion consists of all the data areas, possibly underlined, enclosed in brackets, displayed in high intensity or inverse-video, or otherwise enhanced to show the location and extent of each input field.

In effect, a formatted screen is a giant reusable punch-card with built-in field descriptors. Used for both data entry and information retrieval, it gives the operator the impression of looking directly into the computer's memory.

Specialized Terminals?

Practically any terminal with cursor addressing can be used to display a form on the screen and read or write data to and from selected data areas within that form. Both of the classes of terminals manufactured by HP were suitable candidates, as were most non-HP terminals.

Not all terminals provide the same **visual** effect, but functionally most are interchangeable. For example, differing brands of terminals may offer differing display enhancements, or none at all. At worst, an underline on one may appear as inverse video on another.

Some terminals, including the HP2621, do not even offer a protected screen capability. As a result, you generally cannot erase the foreground without also erasing the background. On such terminals a formatted screen facility would have to clear foreground either by erasing one field at a time or by clearing the whole screen and redisplaying the form. Both of these options are relatively inefficient, and visually annoying as well, though it might still be an improvement over some other method of data entry.

An almost hopeless situation exists with non-protected-mode terminals when there is no means of disabling scroll-mode. Pressing the LINEFEED or RETURN key when the cursor is on line 24 will force the form and its contents to jump up one line, causing misalignment of subsequent fields until the form is redisplayed. Some terminals have a manual

switch for disabling scroll mode, but this is probably not acceptable either.

In summary, a good terminal for formatted screens is one which offers cursor addressing, protected fields and one or more display enhancements. Reliable terminals with these features can be purchased for under $600 apiece. More expensive terminals are available, offering such options as local memory (for storing more than one form), logic for local field editting, and block-mode transmission capability.

Block-mode vs. Character-mode

Within the HP3000 community there seems to be considerable confusion as to the fundamental differences between block-mode and character-mode. For many users, the terms "block-mode" and "formatted screen" are one and the same, while "character-mode" is just another name for scroll-up question/answer dialogue.

Behind these misleading half-truths is a movement I have labelled the **"block-mode conspiracy"**. This subconscious campaign of misinformation promotes the view that block-mode is inherently superior to character-mode, that character-mode is archaic, inefficient, and difficult to use, and that any application which operates in character-mode cannot involve formatted screens.

Granted, block-mode is not really used for much besides formatted screens; and thanks to VIEW, most of the formatted screen applications running on the HP3000 today probably do operate in block-mode. And it is true that character-mode is the only means of implementing scroll-up applications. However, that doesn't mean that character-mode is incompatible with formatted screens. Quite the contrary!

Imagine two identical HP terminals sitting side by side, the one on the left operating in block-mode, the one on the right in character-mode. The forms displayed on the two screens are identical, with the cursor positioned at the beginning of the first input field on each screen.

As you enter the requested information at the left terminal (block-mode), each character typed is displayed on the screen and simultaneously recorded in the terminal's memory. By means of the TAB key(s) you move the cursor from field to field, entering or modifying the values on the screen. When you are satisfied with the contents of memory (as reflected on the screen), you press the ENTER key. This causes the information stored in the terminal's memory to be transmitted to the computer as a block, hence the term block-mode.

Although the casual observer might not notice the difference, something fundamentally different would occur if you were to enter the same information at the other terminal (character-mode). As the name indicates, each character typed would be transmitted to the computer individually, at the instant you struck the key. As soon as the computer received the character, it would be echoed back to the terminal and displayed on the screen.

Whether transmitted as a block or character by character, the data is actually received by the computer one character at a time (two at a time on the model 64) and collected in a buffer in main memory. The difference is that data from a block-mode terminal is presented to the program as one continuous string of concatenated input fields, whereas data from a character-mode terminal is presented to the program one field at a time, as soon as the RETURN key is pressed or the field is completely filled in.

Under block-mode, any processing of the individual fields must be deferred until the whole screen has been entered. When an error is detected, a message is usually displayed in a predefined area of the screen and the user is expected to remedy the situation by modifying one or more fields on the screen. The contents of the screen are then retransmitted to the computer and once again processed by the application program. This sequence continues until no more errors remain.

The technique of processing the whole screen at once may be adopted under character-mode, as well, though it is probably more useful for the program to process individual fields as they are entered, so as to provide immediate feedback to the user. In this way warnings can be evaluated and erroneous fields retyped while they are still fresh in the user's mind.

It also makes sense to automatically align decimal points, expand coded values, calculate sub-totals, or display auxiliary fields directly following each input instead of waiting until the end. For example, if a part number is entered and the program validates the number by retrieving the associated parts record, it is a simple matter to immediately display the description of the part for visual verification.

In either block- or character-mode, default inputs may be predisplayed on the screen, allowing users to confirm a given value merely by TABbing past the field (block-mode) or by pressing the RETURN key (character-mode). This technique saves key-strokes and takes much of the drudgery out of data entry. It is especially helpful in character-mode, where subsequent inputs can be more accurately predicted based on the values of already entered fields.

Using this technique, a medical billing clerk might enter a code identifying the type of service performed and the program would respond with the standard amount charged for that service in 9 out of 10 cases. Only in the exceptional case would the billing clerk have to override the default value. Since character-mode programs ask for data one field at a time, even the sequence of input can be adjusted dynamically in accordance with previously specified values.

A personnel application might have one set of fields for hourly employees, another set for salaried employees and a number of fields which apply to both. Instead of designing two separate formatted screens, all fields could be incorporated into a single form and the program could determine which fields to skip by looking at the value entered for EMPLOYEE-TYPE. With block-mode, the **operator** (rather than the **program**) would have to be programmed to skip the unnecessary fields under the appropriate circumstances.

In effect, each of these techniques provides one more way for the program to help the data entry clerk do a better job. As a result, data entry speeds increase, error-rates are reduced, and operator morale is improved.

Considering that character-mode is inherently more interactive than block-mode, and that this interactivity provides a number of benefits to the user, any effort to discourage the use of character-mode could be regarded as an assault on the users' freedom of choice.

What Effect on System Throughput?

Since the computer is being utilized to do some of the thinking for the user, one might conclude that character-mode would be less efficient, that it would put an additional load on system resources. In practice, the contrary seems to be true. One possible explanation is that character-mode causes the computer to do more productive work while block-mode causes it to do more **un**productive work, or work it wasn't designed for.

MPE's terminal i/o module, for example, was designed to handle randomly distributed single-character interrupts. Block-mode data is transmitted in rapid bursts as much as 1000 times maximum typing speeds. By statistical odds alone, transmission errors and data overruns are far more likely to occur with block-mode i/o than with character-mode i/o.

In addition, the use of block-mode requires every foreground character on the screen to be transmitted even if only one field is needed. Thus the whole screen is likely

to be retransmitted once for each error detected. Not only retransmitted, but probably completely reedited and verified as well.

In character-mode, on the other hand, leading zeroes and trailing blanks don't have to be transmitted, values that are already correct can be confirmed by a single RETURN character, and fields that don't apply can be skipped entirely.

Block-mode applications typically require larger buffers, have more memory, and often impose greater demand on other critical system resources. This is especially true of VIEW applications. As a rule of thumb, you can usually double the number of concurrent users of a program just by switching from VIEW to a good character-mode terminal handler.

Access From Application Programs

The need for HP to provide a formatted screen facility was largely due to the absence of any screen-oriented commands in the existing programming languages. Traditional languages such as COBOL and FORTRAN, invented during an era of all batch processing, never anticipated sophisticated screen interaction. Even BASIC, which was designed for interactive programming, provides for terminal i/o only in scroll-mode.

The ideal solution may well have been to extend these languages to accomodate the technological realities of our day. Unfortunately, COBOL and FORTRAN are the jurisdiction of international standards committees, and a vendor only extends such languages at the risk of an incompatability with the standard sooner or later. There is considerably more freedom with BASIC, and some of the more recent implementations of BASIC do contain screen-oriented constructs for cursor-addressing, etc.

Most commercial applications were still programmed in COBOL, though, and there would have been widespread resistance to any suggestion that the tradition be changed. Any effort to convert established programmers to some new programming language would likewise have caused considerable havoc. In any case, developing a new compiler or rewriting an existing one would have been a very major undertaking even for HP.

In this respect, block-mode offers one distinct advantage. By treating the entire screen as a single block of data (a giant card-image, in effect), traditional batch-input methods could still be applied. Existing batch programs could thus be converted to "on-line" programs with minimal rewriting of program logic. This may account for

the popularity of block-mode among those who have responsibility for program maintenance and development.

Another option HP might have considered would be to develop a self-contained formatted screen utility for data collection and retrieval that operates independently of application programs. For example, QUERY might have been expanded to include a formatted-screen mode.

The problems with this approach are: (1) the vendor tries to anticipate every possible requirement and somehow provide for it, but invariably something is left out, or (2) the resulting facility is so complex that you end up having to learn a special-purpose language in place of a common programming language, or (3) the facility is so rudimentary that separate programs still have to be written to process the data that has been collected.

Was the Project Economically Feasible?

Whatever technical design HP might finally settle upon, it is fairly obvious that a sizeable development effort would not have been undertaken without a marketing strategy for recovering those costs through increased future revenues. Some combination of the following points would undoubtedly have been used to justify the project:

- o make the HP3000 more attractive so more people would buy it (recover development costs through **new computer orders**)
- o make the HP3000 easier to use so existing customers would expand their use of the system and purchase add-on terminals, memory, etc. (recover the costs through **add-on orders**)
- o make the data entry facility so good that HP3000 users would pay a lot for the product itself (recover costs through **software sales**)
- o make the product dependent on a particular type of terminal (and either sell more of these terminals or increase the profit margin on each, or both (recover costs through **terminal sales**)
- o don't worry if the product is inefficient, because users will buy more memory or a bigger computer to handle the volume (recover costs through inflated **hardware orders**)

The last point is included not by way of accusation but to illustrate that the users' interests are not always parallel with the vendors' interests. In the long-run, however, any attempt to take advantage of the user or to

introduce an inferior product would be counterproductive since it would encourage other vendors to provide a better solution.

HP's failure to provide character-mode formatted-screen support, for example, has resulted in a proliferation of terminal handlers, both custom-made and commercially available, with support for non-HP terminals as well as HP terminals. At the opposite end of the spectrum, a number of self-contained data entry products are also on the market.

Users Must Make Informed Choices

Users should be encouraged to use the facility which most often meets their needs and suits their circumstances. Unfortunately, most of us are not sufficiently informed as to the pros and cons of the various alternatives to really make a meaningful evaluation. It is a curious fact that many far-reaching choices have been based on short-sighted ideas and misinformation.

For example, many users will readily admit they don't like using VIEW, but if you ask them why they use it, you'll get answers like:

- "Because we have to have formatted screens"
- "Because it was free"
- "The boss didn't want to waste our block-mode terminals"
- "There's no alternative...it's the only thing HP offers"
- "We thought block-mode would be more efficient"
- "We had no choice...the application packages we bought were written in VIEW"
- "Our salesman says character-mode is on the way out"
- "The training sessions made it sound so easy to use"
- "We just didn't know any better!"

In light of these comments, we would all do well to remember this piece of advice:

Be careful when choosing the tool you'll be using,
So you still can rejoice when there's no longer a choice.

Standards are Necessary

What HP3000 users need is an objective industry standard against which all data entry facilities can be measured and their features compared. V/3000 is the de facto standard for block-mode, but V/3000 lacks most of the features which make character-mode screen form handlers so powerful. The list of design goals proposed in a recent article in the SuperGroup Newsletter might be a good starting point for defining a universal standard.

Users also need some objective performance measurements by which to compare the run-time impact of the various facilities. In 1982, BARUG (Bay Area Regional Users Group) commissioned a Data Entry Task Force to obtain such statistics for the various data entry techniques; as of this date, their report has not been published.

Until standards are established and performance measurements made public, uninformed users will continue to feel their **input is being ignored**, circumstances are conspiring against them, and they are being deprived of those illusive **"systems defined with the user in mind"**.

IMPROVING BACKUP PERFORMANCE--A MODEL

James A. Jonez
Hewlett-Packard Company

ABSTRACT

System backup is one of the most frequently used functions on a typical HP 3000 system. While it is a necessary part of maintaining the system, there has not been much effort in the past to understand the internal interactions among the tape drive, the CPU, and the operating system to assess the limitations and opportunities for increasing backup performance.

Hewlett-Packard has developed a model to predict backup performance under a variety of circumstances. This paper presents how the model has been used to evaluate changes to the STORE program in the Q-MIT and to identify possible changes for future enhancements. It begins with a review of how the tape drive works and how the system accomplishes a typical backup. After the model is presented, the results with the new STORE program are discussed. These results have been verified on an actual HP 3000 system with the 7976A tape drive. Finally, the use of the model for future backup performance improvements will be addressed.

I. INTRODUCTION

This paper, while it describes a model for backup performance on a simple system configuration, will help the reader understand the process of backup and those factors that actually affect the backup time for any system configuration. The paper covers some general concepts relating to tape peripherals and how they work as well as how the rest of the system interacts. The reader will also gain an insight into the directions HP is taking for backup in the future.

Backup performance is a function of many system parameters and not just the tape drive specifications. Many times the tape drive specs are used out of context of true backup performance. System interactions involved in a backup are very complex. HP developed this model to propose modifications to the system to improve backup performance. With this model HP can provide better solutions in both hardware and software than are currently available.

II. CONCEPTS OF TAPE DRIVE TECHNOLOGY

Some background information will help to explain the operation of the system during backup and the way the model works. Backup provides an insurance policy against a system failure or a user error that would destroy some data stored on the system. Basically, it involves the transfer of data from the primary mass storage device, usually a disc, to a secondary mass storage device which is most frequently a tape drive. Tape has some obvious advantages for off-line storage since it is very reliable, relatively inexpensive, and can store a large amount of data in a small space. The parts of the system involved in a backup transfer include hardware (the disc drive, CPU, cables, and tape drive), utility software to manage the data transfer (STORE or SYSDUMP in MPE), and the operating system which has the software drivers for the peripherals.

RECORDING DATA ON TAPE:

A brief review of the data structure on tape will clarify the comparison of different types of tape drives later. Half inch reel to reel tape drives have been around for a long time. The data standards have been well defined and well accepted in the industry. Most tapes in use today are recorded in nine parallel tracks. The tape head has nine tracks that can read or write on the tape simultaneously. The data density on the tape can be measured in bits per inch or BPI for each track. One of the nine tracks is used as a vertical parity check so that the other eight tracks can record one byte of data in the length of each bit cell. Thus the tape density connotes bytes per inch as well as bits per inch. The density is often expressed as logical characters per inch, or CPI, to distinguish the data density from the density of the actual magnetic flux reversals that are recorded with the different formats.

The three industry accepted formats that exist today are 800 CPI Non Return to Zero Inverted (NRZI), 1600 CPI Phase Encoded (PE), and 6250 CPI Group Coded Recording (GCR). The algorithms for converting logical ones and zeros to magnetic flux reversals are specified by the codes NRZI, PE, and GCR. More complete descriptions of these formats can be found in many publications.

For all of the formats, the data within a file are divided into records. Records are typically 4 Kbytes, 8 Kbytes, or 16 Kbytes. The record size is specified by the host. The end of each file is marked with a special type of record called a file mark. File marks and records are both called blocks.

Between each block is a blank space on the tape called an inter block gap or IBG. The nominal length of the IBG is 0.6 inches in the 800 CPI and 1600 CPI formats and 0.3 inches for 6250 CPI. The ANSI standards that define these formats allow the gap length to extend up to several feet.

A transfer from the host to the tape peripheral involves three steps. First the host must send a command to the tape drive controller to tell it what to do such as a command to write a record of data. Next the data must be transferred and finally an acknowledgement is sent back from the tape drive to the host to indicate that the data were written correctly on the tape. This command, transfer, and response sequence is repeated for each block.

START/STOP AND STREAMING TAPE DRIVES:

A start/stop tape drive has the capability to stop and start the tape motion within the length of the IBG. Since the gap is rather short, these drives employ tension arms or vacuum columns to "buffer" the tape while the capstan controls the actual tape speed across the head. This allows the tape drive to stop the tape in the IBG, send the acknowledgement back to the host, wait for the next command, and then respond quickly for the next block. On high speed start/stop drives this operation takes only a few milliseconds.

Figure 1(a) on the next page shows what happens to the tape motion across the head. At the top is a piece of tape showing the IBG between two blocks of data. The graph below shows the velocity of the tape relative to the head. The tape moves at a constant velocity as it reads or writes the block of data then stops in the IBG and waits for the next command. When the command is received, the tape speed ramps up to read or write the next block. The command, transfer, and response sequence must be completed for each block before the next sequence begins.

Recently, the traditional start/stop tape drive designs have been challenged by a new type of drive called streaming drives. Simply put, streaming describes the operation of a tape drive when data are supplied or requested such that the tape does not stop in the IBG. Figure 1(b) shows the tape motion in streaming mode. Note that the tape does not stop at all in the IBG. One advantage of a streaming drive is that this saves the time required to stop the tape and start it up again. Tape drive manufacturers also point out that a streaming drive does not need to stop and start rapidly so that the tension arm mechanisms or vacuum columns needed for tape buffering can be eliminated resulting in a much lower cost product.

However, there are some problems with this streaming

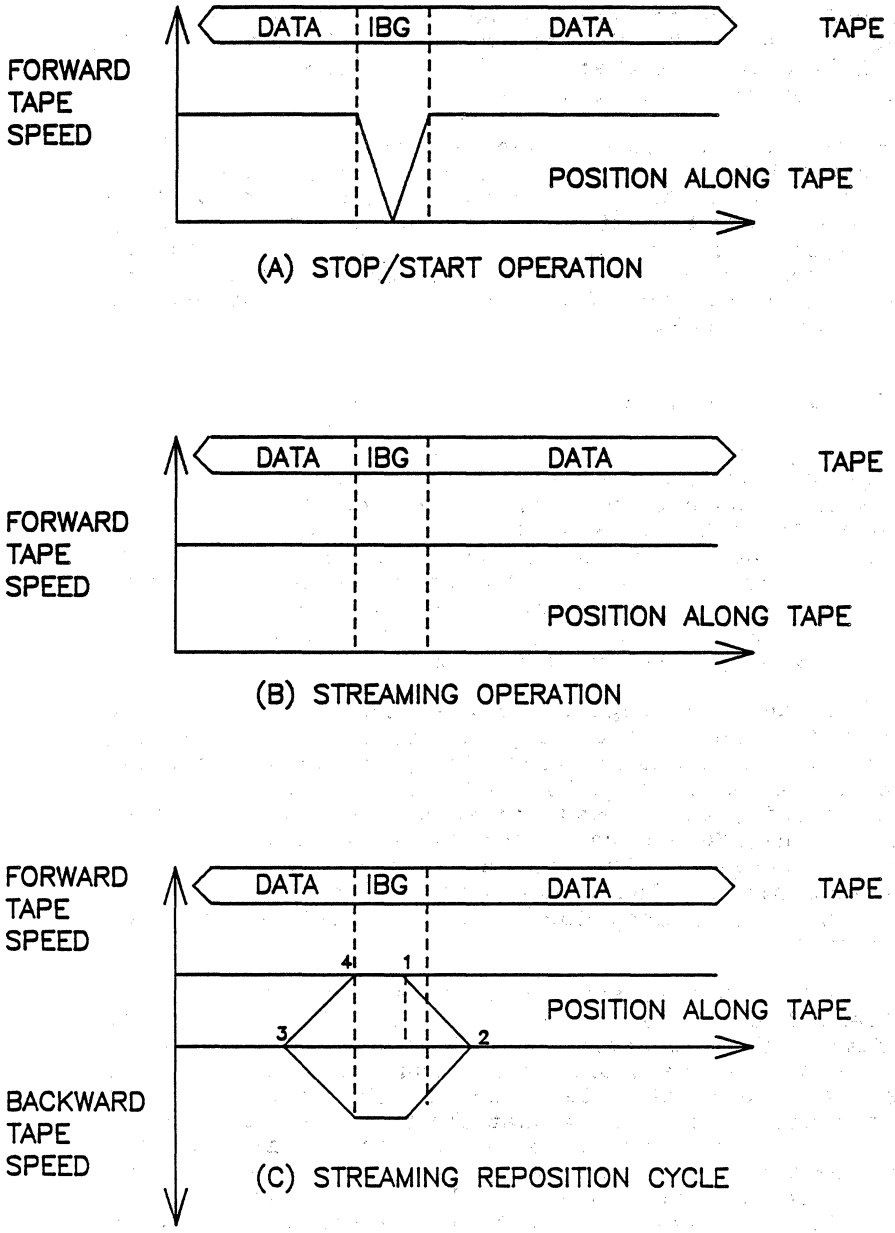


FIGURE 1
TAPE DRIVE OPERATION

technology. Figure 1(c) shows what happens if the streaming drive is forced to stop. The command for the next block must be received slightly in advance of that block so that the tape drive can respond to the command. If the next command is not received by the time the head is positioned at point 1, the tape will slow down and stop at point 2. Since this is beyond the start of the next block, the tape must be reversed to point 3 where the drive will wait for the next command and ramp up to full speed at point 4. The time penalty for this reposition cycle can be a half second or more. Most host systems do not support streaming tape drives. At 100 inches per second tape speed the 0.6 inch gap is passed in only 6 milliseconds--not much time for the CPU to respond to the acknowledgement from the last record, send the command for the next record, and begin the data transfer. As a result, streaming drives on most systems end up not streaming at all.

To get around this problem, HP designed the software driver in MPE to work with the tape drive controller to accommodate this streaming mode operation. The 7976A tape drive is a high speed start/stop drive that is also designed to run in a streaming mode. This is coordinated with the host (particularly the I/O tape driver and the STORE program) so that they operate as a system. This is done with a technique called command queuing. Even before the acknowledgement from the last (or current) record is sent back to the host, the host can send the next command which is stored in a special command queue buffer in the tape drive controller. When the tape head reaches the IBG, the command for the next block is immediately available so that the tape will not stop in the gap. In the proper time the acknowledgement from the last block will be sent back to the host so that the command, transfer, and response sequence for adjacent records can be overlapped. With this command queuing technique, the 7976A tape drive will operate in a streaming mode on HP 3000 systems.

BACKUP PERFORMANCE:

The measure of backup performance is often misunderstood. The most frequently quoted specification for tape drive performance is the burst transfer rate. This is defined as the product of the tape speed and the recording density. For a high speed tape drive a typical burst transfer rate is 75 inches per second times 6250 bytes per inch or 468.75 kilobytes per second. However, the tape drive is only one part of the system involved in backup. While the burst transfer rate is an accurate measure of the upper limit of data transfer to the tape, it is a poor indicator of backup speed since it does not take into account the operation of the system or even the data structure on the tape.

The burst rate is applicable only within the data record on the tape. With the 6250 CPI format an 8 Kbyte record is 1.28

inches long. At 75 ips, this takes only 17 milliseconds to transfer. Each record is separated by a 0.3 inch nominal IBG so the time to pass the record plus the gap is 21 milliseconds. These gaps alone will reduce the transfer rate to 8Kbytes/0.021 seconds or about 380 Kbytes per second. Using 16 Kbyte records the actual transfer rate will be a little better, but longer gaps, shorter records, file marks, and the operation of the system will all reduce the actual transfer rate even further.

At the other end of the spectrum is the system effective transfer (SET) rate. This is calculated simply by dividing the amount of data transferred by the time it takes to transfer. This takes all of the system effects into consideration and gives a true representation of the actual backup performance on the system.

The best performance test of any peripheral is the test on your particular system. Since backup performance will be affected by the type of system, the configuration, the software, and even the file set characteristics, the choice of a benchmark is important. In general, the relative performance for different tape peripherals will change depending on these factors. To establish a valid benchmark, HP has collected file information from many systems representing over 6 Gigabytes of data to characterize a typical file set.

III. PRESENTATION OF THE MODEL

It is difficult to separate and analyze all of the parameters that affect the actual backup time. This is where a model is useful--to give a relationship between the system parameters and actual backup performance. The model describes a simple system configuration and uses the characteristics of MPE.

The backup model simulates a system to find and isolate sensitivities in the system. It was used iteratively to first identify parameters with the highest sensitivities on backup performance and then move in a stepwise fashion to analyze the other parameters. A basic assumption is that the sensitivity at the initial branches is higher than at the lower levels. Interactions among the parameters were assumed to be small. As it turns out, there are a few parameters that control most of the sensitivity to backup performance. These are: the file structure, disc access times, host buffering technique, and the tape speed and density.

The model uses a straight forward approach. Most of the parameters are not treated as random variables but are simplified by using average values such as disc access time, system overhead, and others. This results in a model that

will quickly identify the most sensitive areas. As we learn more about the entire process the model can be developed to reflect more detail so we can finely tune the system.

Figure 2 on the next page shows a block diagram of the elements in the system that are included in the model. Starting with the disc and the interface, the data are fed into the HP 3000 through the GIC (General I/O Channel). The STORE program controls the file transfer operation within the CPU. Data are sent through another GIC to the tape drive and stored in a local buffer there. The formatting electronics convert the data bits to the flux reversal pattern recorded on the tape.

This diagram can be used to demonstrate how the tape drive burst transfer rate differs from the SET rate. The burst rate is the transfer rate from the tape formatter to the tape for data within each record. The SET rate includes all of the factors on the diagram.

Since the file structure has a large affect on backup, several actual file sets were used as inputs to the model. An input file set was generated by first doing a LISTF(fileset)-1 to produce a tape of all file labels. LISTF and STORE both make use of the same sorting algorithm so the LISTF output of file labels is in the same order as the STORE program. A program called MAKEGF was then used to create a sample Goodfile with the following information: file name, account name, group, user name, and all of the extent sizes and addresses. (The Goodfile is used by the STORE program and contains the names of the files to be stored.) This sample Goodfile becomes an input file for SIMSTORE which is the actual simulation program. The program produces detailed and summary outputs as well as typical system comments. A block diagram of this process is shown in Figure 3.

Fixed parameters in the model defined the hardware configuration. The system was a Series 64 or 44 (the difference in overhead was considered negligible), with one 7933 on a GIC and a 7976A on a separate GIC. The STORE was run with no users on the system except the System Manager. Variables were based on the file structure, the HP 3000 STORE program, the disc drive and the tape drive. Table 1 lists all of the parameters in the model under each group.

IV. THE RESULTS

The simulation results agreed very closely with the actual results measured on a system using the same file set. A comparison of the times for a STORE using two different file sets is shown below:

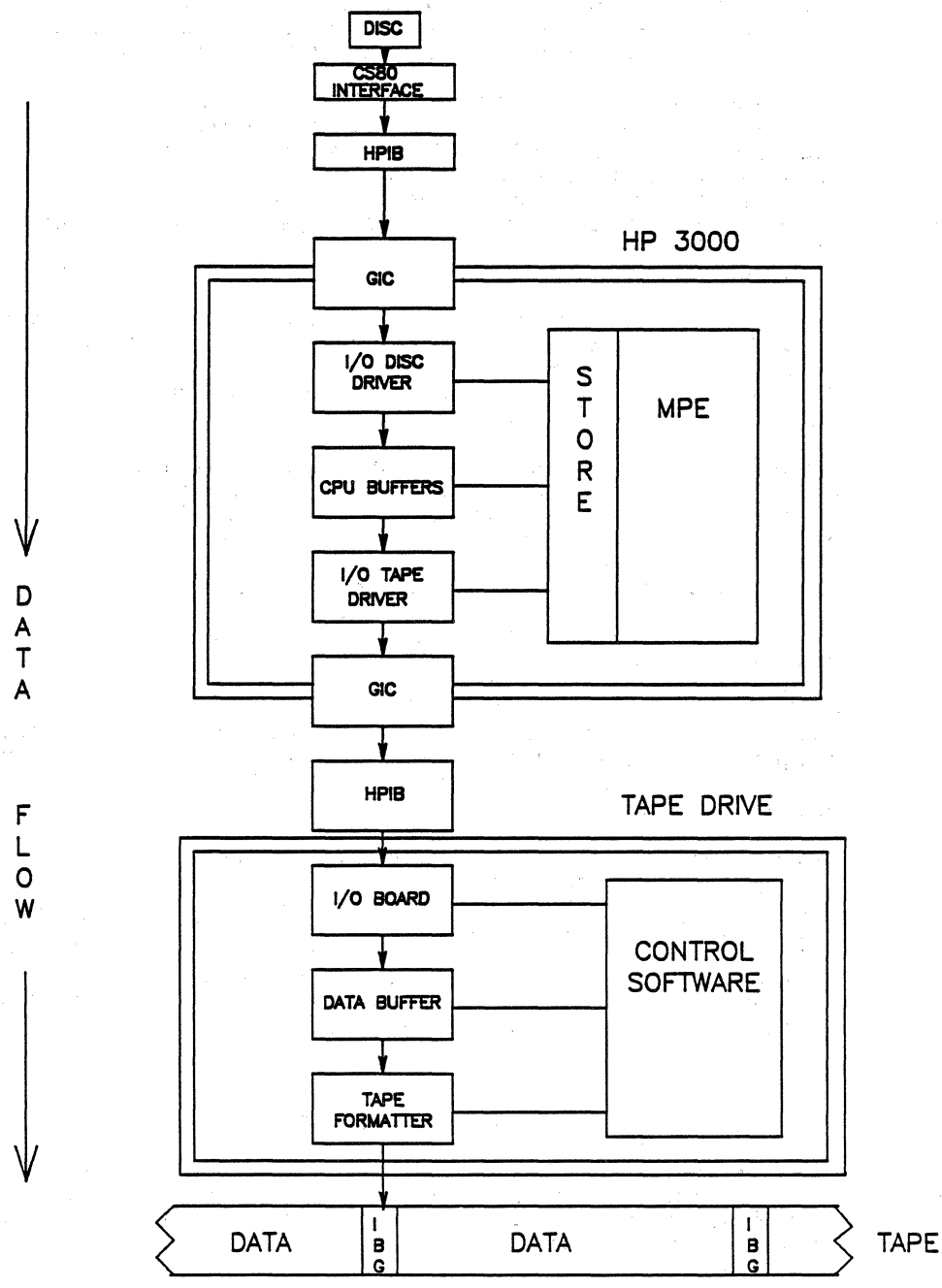


FIGURE 2
BLOCK DIAGRAM OF SYSTEM

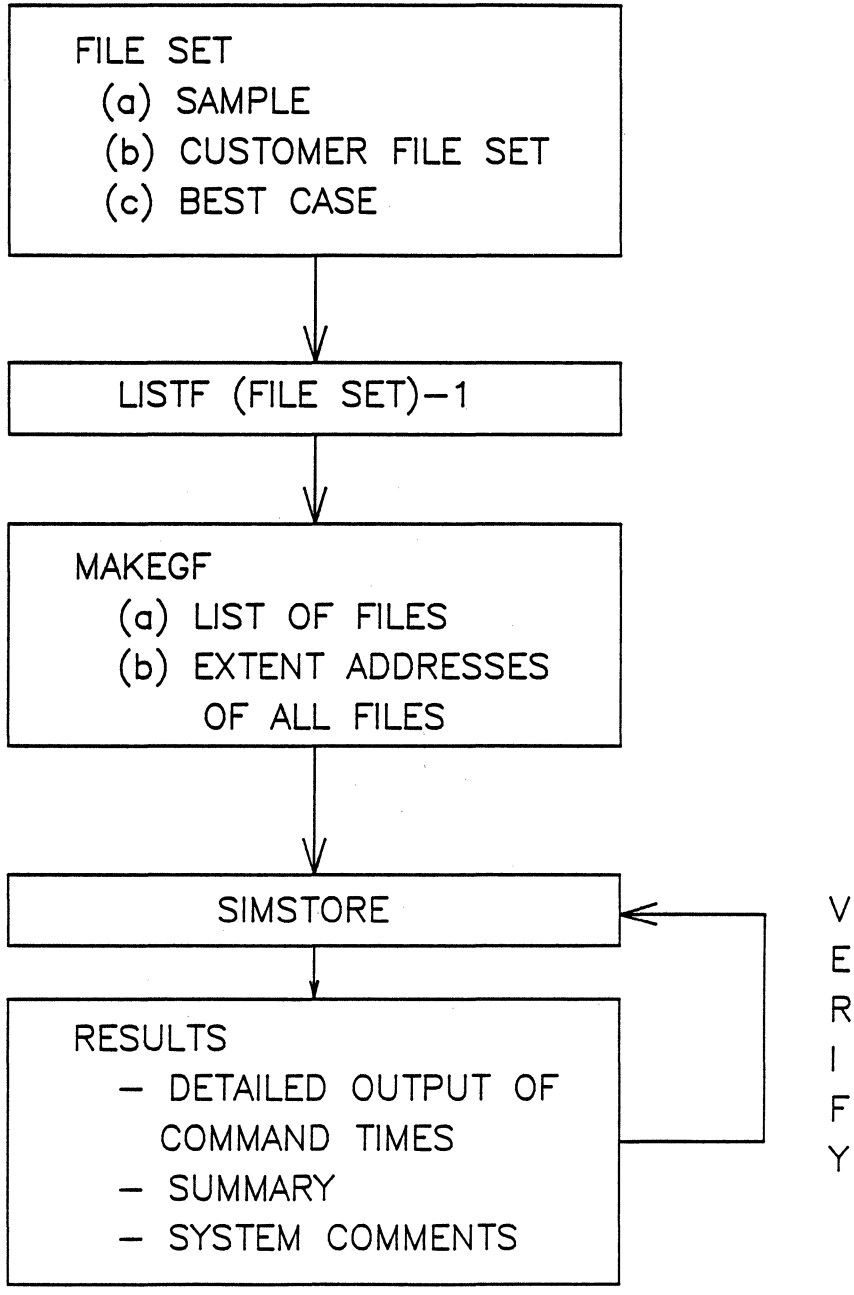


FIGURE 3
BLOCK DIAGRAM OF MODEL

TABLE 1

Parameters Used in the Model

Tape Parameters.

- Tape speed
- Tape format (density)
- Record size
- IBG length
- Buffer size
- Command queuing

Disc Parameters.

- Burst transfer rate
- Access time.
- Latency
- Seek time
- Bytes/track
- Sector size
- Command set 80 overhead
- Buffering
- Disc type

HP 3000 STORE.

- Goodfile
- Users on system
- File boundaries
- Overhead
- I/O channel program
- I/O command protocol

File Set Structure.

- Extent size
- Number of extents
- Blocking factor
- File size
- Locality of extents
- Label

File set	Actual Store (Elapsed time)	Simulated Store (Elapsed time)	% Error
@.Kevin.Account	2:29	2:32	2.0%
@.@.@	7:11	6:56	3.5%

Additional testing has confirmed the accuracy of the model.

To understand how the tape drive will affect the backup time several different types of drives were included in the model. These tests were run using the version of the STORE program that is available in the Q-MIT (MPE release C.01.00) and the system configuration shown in Figure 2. First, the upper limit to tape drive performance on the system (in this configuration) is shown by assuming no tape drive at all. This is simulated by forcing an immediate response from the tape drive. This case models the disc access process, the overhead of the Goodfile, the STORE program, the CPU buffering scheme, and the GIC. It does not include the electronics or mechanics of the tape drive. The resulting performance limitation to the system alone was 325 Kbytes/second.

Next, by adding the parameters of a tape drive into the model the performance can be estimated in an actual backup situation. Using a 125 ips start/stop tape drive under the same conditions, the transfer rate remains at 325 Kbytes/second which is equivalent to an average tape speed during the STORE operation of 76.7 ips with the 6250 CPI GCR format. The effective tape speed is reduced compared to the rated tape speed of the drive because of the data formatting and the fact that the drive may wait on the system.

Another simulation was run using the parameters of a 75 ips start/stop drive. In this case the throughput dropped about 9% to 297 Kbytes/second. The average tape speed during backup was 68.7 ips.

Some testing was done to learn how the buffer size in the tape drive would affect performance. Host buffer size was held constant at 8 buffers of 255 sectors each while tape record size and tape drive data buffer size were varied. (Note that over 500 Kbytes of memory were allocated to buffering in the host.) Host buffers were released when the tape record had been verified. Even if the host buffers are released as soon as the data are transferred to the tape drive, only a slight performance improvement can be realized by increasing the buffer size in the tape drive. The explanation is that more buffers will help when the transfer rate through the system is close to the transfer rate onto the tape. As it is today, the burst transfer rate to the tape is much higher than the transfer rate through the system so the additional buffering in the tape drive has only a small effect. More study is needed in this area to verify these early results.

The STORE program has a very large affect on backup performance. Two key factors are the file boundary overhead and the host buffering algorithm. With the version of STORE that is widely used there are four to six disc I/O's at each file boundary. This has been changed with the new version of STORE available in the Q-MIT to only one or two disc I/O's. This is done by reading the file label at the same time as the data are read from the first extent and improving the method of keeping track of which files to write to tape. In addition, the program will now read contiguous extents from the disc with a single disc I/O. The buffering scheme has also been improved. Instead of two 24KW host buffers, STORE now uses three 16KW buffers. Without increasing the total amount of memory allocated to buffers, the performance can be improved substantially.

Some interesting statistics have been compiled based on the file label information from the systems that were sampled. Information on file sizes and extent sizes are particularly valuable since these can have a large affect on backup performance. The graph in Figure 4 on the following page shows the cumulative extent and file sizes as a percent of the total. The graph shows that the median file size is 33 sectors and the median extent size is 14 sectors. Further analysis has shown that 10% of all extents hold 75% of all data and just one percent of the extents contain 35% of the data!

Figure 5 shows the cumulative percentage of all files versus the number of extents per file. This is the percentage of files with X or fewer extents. Note that 65% of all files have only one extent and that over 80% of all files have eight or fewer extents.

The effect of the file sizes and extent sizes can be dramatic. Because of the system overhead associated with each file or extent and the disc seek time due to disjoint extents, backup performance can be increased by increasing the size of files and reducing the number of extents per file. A performance improvement of 11 to 17% is possible if all files have only one extent instead of up to 32 extents. The small extents are the ones that cause the most trouble during backup because of the overhead involved. While 50% of all extents are 14 sectors or smaller, they contain only 7% of all data. Most of the data on the system are packed into the larger extents so that only a small part of the data on the system can cause the backup performance to be poor. System managers can also improve performance by doing a RESTORE to clean up the data structure on disc.

V. SUMMARY

Hewlett-Packard is committed to better backup solutions. The

CUMULATIVE EXTENT AND FILE SIZES AS A PERCENT OF TOTAL

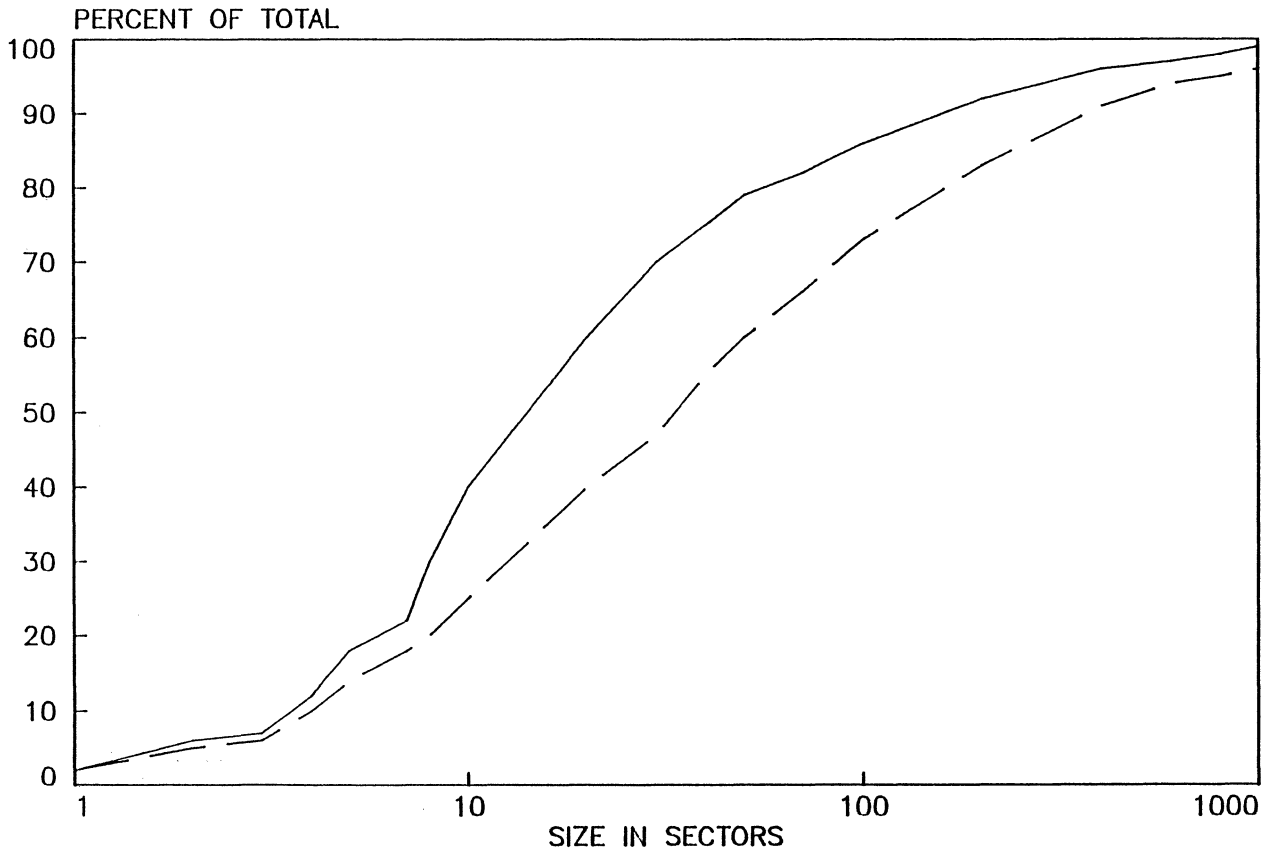


FIGURE 4

CUMULATIVE PERCENTAGE OF ALL FILES VERSUS NUMBER OF EXTENTS PER FILE

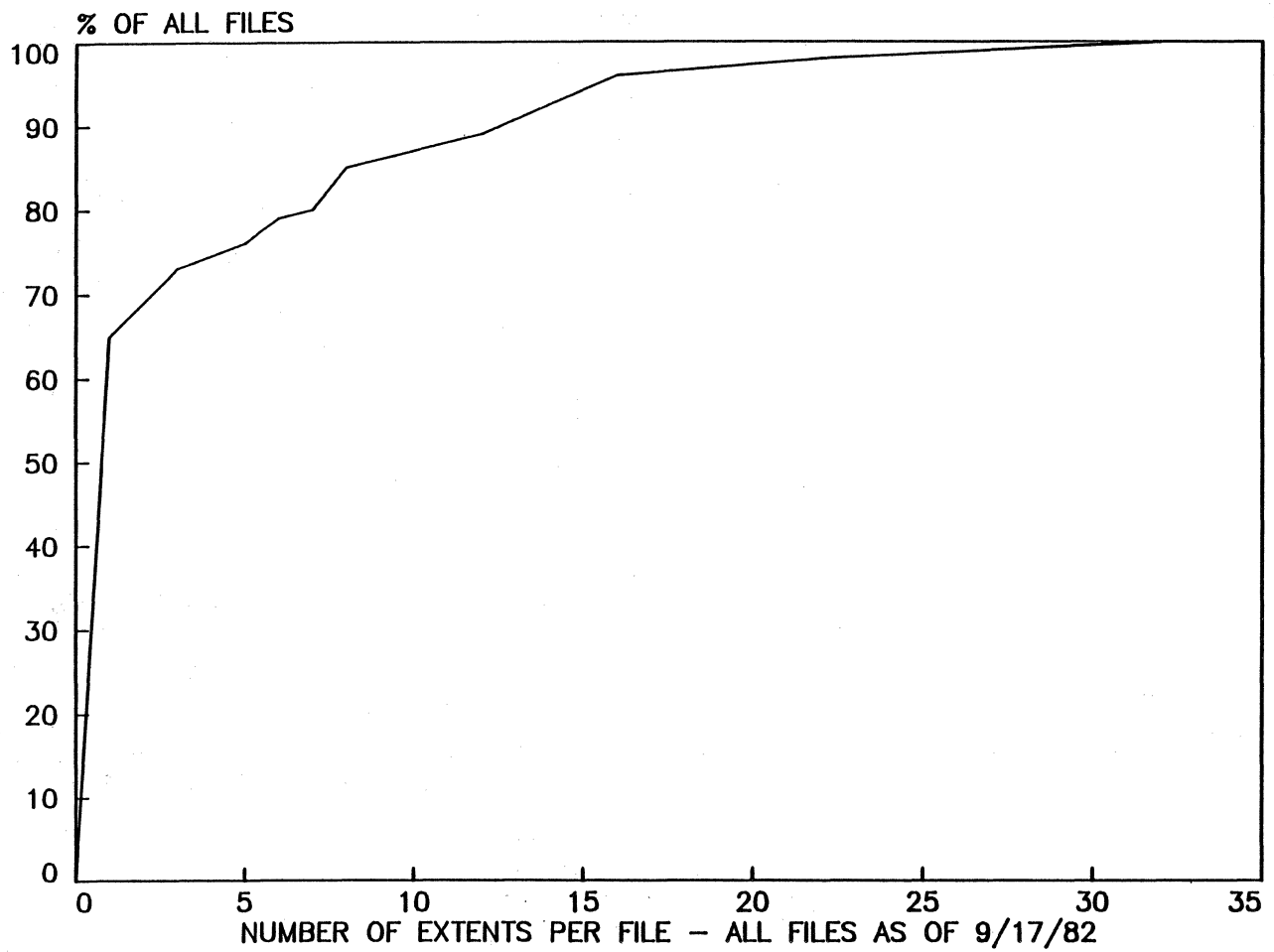


FIGURE 5

work that HP has done to model backup performance has already resulted in some changes to the STORE program as part of the Q-MIT. The reduced overhead for extent boundaries and better buffer allocation will help reduce backup time on most systems.

HP will continue the work that has been started to understand backup more fully. Future developments at HP will include possible changes to the hardware and software to make the tape drive work as an integral part of the system. HP is investigating enhancements to the operating system, different tape drive designs, and more software to provide a backup strategy which addresses higher performance and greater flexibility. The overhead in the system may be reduced even further and we have found a better method of buffering data in the host that will be available in the near future. New designs in the tape drive will provide greater reliability with lower cost products. The electronics in the tape drives will become more sophisticated and allow better data handling. Finally, some totally new approaches to backup such as an image copy may be a good solution to some HP 3000 users' needs.

OPTIMIZING TRANSACT CODE

Robert A. Karlin
Consultant

INTRODUCTION

In 1978, 20th Century-Fox Films embarked upon a project to connect its 21 branch locations into an on-line network for the booking and billing of its feature films. After a long vendor selection process, a pilot project was begun using the HP3000. In our search for a user-friendly environment we chose to use a menu oriented structure written with V/3000. We needed a productivity tool that could handle V/3000 intrinsics and yet be faster and more flexible than COBOL or SPL. We chose IMACS, drawn by its high level IMAGE interface and its dictionary driven language. Our shop was already familiar with Data Dictionary/Data Directory concepts, and felt that the IMACS dictionary would fit with our already developed techniques. Our user wanted to be completely isolated from MPE, and once signed on and into the menu structure, he wished to leave it only to exit completely from the system. Our user also needed the ability to move reasonably rapidly between the different functions of the system, and could not tolerate excessive delay in loading segments of code.

The system has been in operation for somewhat over a year at this point, and is about 80 percent complete. It consists of over 150,000 lines of TRANSACT code and is written as a single segmented program. Our response time in moving from application to application is under 5 seconds. Our system is now running on three HP3000/64s with about fifty five to sixty users per machine. In our development effort, we have probably exceeded every limit of the language. This paper is a compendium of odds and ends learned during our development project.

I have divided the subject of TRANSACT optimization

into two broad areas, stack size considerations and run time speed. There are tradeoffs between these, and where there are conflicts, I have attempted to identify them. At the end of this paper I have included a few notes about INFORM and REPORT.

STACK SIZE CONSIDERATIONS

TRANSACT

Transact is an interpreted language. The TRANSACT compiler reads your source code and produces a file of TRANSACT P-code(pseudo-code). TRANSACT P-code is a shorthand form of your source code, and consists of a series of tables containing the descriptions of your data and source, and a coded form of your source. The TRANSACT processor reads the P-code, and interprets the the coded source on an instruction by instructin basis. It is both faster and stack efficient to use a single instruction for multiple operations as the processor only has to interpret the instruction once. For example:

```
LET (FOO) = (BAR) + 5;
LET (FOO) = (FOO) * 1.7;
```

is less efficient than:

```
LET (FOO) = [(BAR) + 5] * 1.7;
```

ITEM DEFINITIONS

TRANSACT keeps a table containing all of the parameters needed to identify and use each data item. As part of this definition, TRANSACT keeps the full item name as it appears in your source code. Since this name is kept in a variable length field, the shorter the name you chose, the smaller this table is. Transact keeps a separate entry in this table for each item referenced, even if it is a redefinition of another item. Even though this seems to use more stack space, you can use this to your advantage.

OPTI

TRANSACT provides an option that can save you a considerable amount of stack space. The OPTI option is specified on the CONTROL line prompt when executing the compiler and has two functions. First, the compiler

will remove from its internal item table any data item definitions not referenced in your code. (DEFINE(ITEM) does not constitute a reference in this case.) This is especially useful if your internal standards require the use of copy books for your item definitions. Second, the compiler will remove the data item name for any item in which the OPT parameter is specified in the DEFINE(ITEM) statement. This can only be done in cases where the data item name is not required to retrieve the item from an image data base. But, the OPT parameter CAN be specified on data item redefinitions of data base data items. For example if your data base contains the item FOO, and this item occurs in your inventory and customer files you may use the following without seriously affecting your stack size:

```
DEFINE(ITEM)
                FOO X(10):
                INVENTORY-1ST-00 = FOO, OPT:
                CUSTOMER-1ST-00 = FOO, OPT;
                      .
                      .
                      .
GET INVENTORY, LIST=(CUSTOMER-1ST-00:);
```

TRANSACT will store only the literal "FOO", and resolve the reference at run time.

LITERALS

The TRANSACT compiler is smart enough to recognize recurring literals and store them only once. Since TRANSACT also allows you to split literals into smaller strings on the MOVE statement (a side effect of allowing you to span lines with a literal) you can save a tremendous amount of space by carefully coding your application. For example:

instead of coding this:

```
MOVE (PAGE1) = "PAGE 1";
MOVE (PAGE2) = "PAGE 2";
```

use this:

```
MOVE (PAGE1) = "PAGE " "1";
MOVE (PAGE2) = "PAGE " "2";
```

In the first case you are storing 12 characters, 'PAGE 1PAGE 2'. In the second you are only storing 7, 'PAGE

12'.

SEGMENTATION

One of the easiest ways of squeezing your application into a smaller size is by segmenting it. TRANSACT segmentation is a single level tree. Your root segment is always present and all other segments are swapped into the same area appended to the root area. All items that are in the root may be referenced by any segment, while all items in an overlay segment can be referenced only in that segment. Any procedure in the root segment can be referenced by anyone, while the only procedure names that can be referenced from an overlay segment must be defined in the root using the DEFINE(ENTRY) verb. The amount of stack space used is the sum of the stack space used by the root segment and the largest overlay segment.

The amount of time nescessary to effect a segment change is relatively small, but not inconsequential. Do not write code that bounces between segments. Change segments only between major functional changes (such as between applications on a menu or between high level($$) commands.

Note that when you are using a segmented program, the processor keeps the p-code file open during the entire duration of the process. This does add a certain amount of overhead.

When using the segmentation feature, you must clean all local items off your list register before overlaying the segment. If you must leave items on the list register (items, not data. You can always remap the list register over the data register after resetting it and not lose the data originally stored there) use the OPTS option on the CONTROL line during execution of the compiler. This will prevent the processor from aborting when you leave the overlay segment.

CALLING

TRANSACT has a feature that will allow the execution of another TRANSACT program that will share the same DATA register as the calling process. This allows the ability to pass data to a TRANSACT subprogram, and overcomes some of the limitations of

segmentation. Since the subprograms are separate programs, they are compiled separately, alleviating the problem of hour long compiles and reams of paper to keep a current listing. Specifying the SWAP option on the calling statement decreases the stack size necessary by allowing the processor to overlay portions of the tables in the calling program.

On the other hand, calling is slow, taking 5 to 10 times the length of time that the segmentation option takes. Using the SWAP option slows it even more due to the time necessary to reload the overlayed areas.

V/3000

If you are using the TRANSACT V/3000 interface, there are certain steps that you can take to decrease your stack requirements.

First, always use a V/3000 fast forms file. This will not only decrease your stack requirements, but will also decrease your disk IO.

When using V/3000, a buffer is set aside for terminal IO. The size of this buffer is dependent upon the size of the largest form in your formfile. Keeping your forms about the same size will decrease your stacksize dramatically.

If you specify a V/3000 formfile to TRANSACT, the processor will attempt to load the complete set of item definitions to describe the formfile. If you are only using a small part of a large formfile, this could be disasterous. Use the VPLS option on the SYSTEM statement to specify only the forms that you will be using in this program.

MESSAGE CATALOGS

HP suggests that one method of reducing stack requirements is to remove all error and informational messages to an external file, referencing this file to display messages instead of using literals. This technique should be used only when the IO overhead is not a consideration for run time performance.

RUN TIME OPTIMIZATION

Even though TRANSACT is an interperative language, most rules of run time optimization are applicable. I will not attempt to recapitulate these rules in this paper. Those interested should refer to Jim May's exceptional paper on PROGRAMMING FOR PERFORMANCE, Bob Green's work on Data Base optimization, and Eugene Volokh's scholarly work on Data Base synonyms. I shall only cover those particular cases that apply to TRANSACT.

ARITHMETICAL OPERATIONS

TRANSACT is not exceedingly fast in mathematic computation, but there are a few things that can be done to speed up your calculations. First, combine as many operations as is feasable into one TRANSACT sentence. The overhead of interperating the instruction can mount rapidly in recursive operations.

Secondly, use either all packed operands, or all integer operands. In any other case, TRANSACT will convert all operands to packed format. This can be especially time-consuming if you are using display type fields.

In any case, if you have complex mathematical requirements, you should consider a FORTRAN or SPL subroutine.

SORTING

Since TRANSACT calls the SORT subsystem to do all of its sorting, do not use the SORT option lightly. Sorting a few items in order to display can cause more system overhead than the use of IMAGE sorted keys to keep these items in order. Note that in order to call SORT, you must have at least 4K words of space free on the top of stack to prevent stack overflow.

PROCEDURES AND SUBROUTINES

In order to call a external procedure from transact, you must place it in an SL. TRANSACT then uses the HP LOADPROC procedure to invoke your procedure. This is very, very slow, especially during the initialization of your program. At TCF, we found that a program calling about 10 subprograms could take as much as five minutes to initialize on an unloaded

HP3000/44, or over a half hour on a load machine. If you use external procedures, combine them into one segment before placing them in your SL to minimize the overhead of loading. Also, if you use few procedures, and reference them rarely, use the NOLOAD option of the PROC verb to prevent issuing the LOADPROC until the procedure is truly needed.

TRANSACT provides a method for eliminating the issuing of the LOADPROC system intrinsics. If you are calling system intrinsics from TRANSACT, you should define them in the DEFINE(INTRINSIC) statement. Note that not all intrinsics are available this way.

SEGMENTATION AND CALLING

Minimize bouncing between called programs, and between overlayable segments. Put your global subroutines in your root segment, since there is much less overhead in performing subroutines in the root segment. And, of course, try and keep your segments of approximately the same size, since TRANSACT will allocate space on the stack for the largest segment.

After you have thoroughly debugged your program, compile it with the OPTS option. This will remove a certain amount of the internal checking when moving between segments. Note that if you have left local items on the list register, you could produce very strange and interesting results.

IO CONSIDERATIONS

When accessing flat files, use the FILE verb as opposed to GET, or PUT. Using the FIND verb with the PERFORM option is more efficient than coding your own construct, though be careful of TRANSACTs internal data base error routines. When using the FIND verb, be extremely careful of the LOCK option. Transact does not lock on the record level. The following construct could put your system to sleep.

```
      FIND FOO, PERFORM=BAR, LOCK;
           .
           .
           .
BAR:
    DISPLAY RECORD-ID;
  PROMPT RECORD-NEWSTUFF("Enter Stuff or RETURN to
```

```
bypass");
      IF RECORD-NEWSUFF <> " "
      THEN
         PUT BLEECH;
      RETURN;
```

The above example will lock your base while waiting for your operator to finish answering the prompt, impeding all other processes waiting for it.

If it is necessary to lock multiple records for update, it is probably more efficient to lock them yourself than to use TRANSACTs locking algorithm.

OFFSETS

Probably the single most time-consuming action that you can do within TRANSACT is LET OFFSET(FOO) = anything. LET OFFSET will recalculate the displacements of the entire record definition. If it is necessary to use arrays, try to construct them in such a way that you limit the number of times that you change the offset into that array. Algorithms that calculate the offset into an array are much faster than incrementing through the array entry by entry. Accesing the elements of the array is reasonable efficient, such that algorithms that can reference multiple entries in the array without incrementing the OFFSET is faster than accessing entry by entry.

V/3000

If you are using V/3000, avoid switching from block mode(GET(FORM), PUT(FORM), etc.) to character mode(DISPLAY, PROMPT, DATA, etc.). Also, avoid using $$ commands and subcommands. Use menu for program option selections instead. Keep your forms reasonable in size to decrease your overhead.

INFORM and REPORT

Many users, enthralled with the idea of producing reports for their users without programming, or allowing their users to access the data in their files without DP intervention, have run into the problem of INFORM or REPORT taking 3 to 4 hours to produce a 10 line report that could be produced in TRANSACT, COBOL, BASIC, or even by hand in 5 minutes. The reason behind

this can be found in the method that these packages access your data. Consider a likely application. You have a data base with a master file containing 1000 entries. Chained to this file you have a detail of 100,000 entries. You wish to search for a range of entries from your master file, and process all entries that are chained to it, a total of perhaps 3000 records. When you write this in COBOL or TRANSACT, you read the master file, select the records you want and read the appropriate chains. INFORM and REPORT, on the other hand, will automatically read the detail file first, and then read the appropriate master. In other words, you will read the whole 100,000 record file first, instead of the small master. One method of circumventing this would be to extract your selection keys to either a flat file or a KSAM file. Since INFORM and REPORT read flat and KSAM files before attempting to process your detail set, your efficiency should skyrocket.

TYPES:
INTERNAL
EXTERNAL

PURPOSE QUALITY CONTROL,EFFICIENCY,SHAREHOLDERS INTERESTS, COMPLIANCE WITH GOVERNMENT REQUIREMENTS

HISTORICAL

FROM MANUAL TO MAINFRAME: LARGE COMPANIES
MAJOR PROJECT
BATCH

THEREFORE
1. CLEARLY DEFINED AUDIT TRAIL
2. CONFIRMED BY AUDITORS BEFORE IMPLEMENTATION
3. IN THE EVENT OF "BUGS", SYSTEM PROBLEMS,SYSTEM FAILURES OR DISCOVERY OF INCORRECTLY ENTERED DATA IT WAS POSSIBLE (ALTHOUGH NOT ALWAYS EASY) TO REENTER THE TRANSACTIONS OR TO CHECK THE ACCURACY AND INTEGRITY OF THE INPUT DATA.

TO A VERY SIGNIFICANT DEGREE AUDIT HAS FOLLOWED COMPUTER TECHNOLOGY. THE APPROACH HAS BEEN TO RESPOND TO INCREASED USE OF COMPUTERS BY INCREASING THE TRAINING (AND THEREFORE COMPUTER AUDIT ABILITY OF STAFF).

CURRENT

THE POPULARITY AND USE OF MINI COMPUTER SYSTEMS CHANGED THE PICTURE QUITE SIGNIFICANTLY:
THE LOW ENTRY PRICES FOR MINI COMPUTER SYSTEMS MEANT THAT COMPUTERS PROLIFERATED THROUGHOUT LARGE, MEDIUM AND SMALL COMPANIES.MANY COMPANIES WITH MINI COMPUTER SYSTEMS HAVE THEIR OWN PROGRAMMING STAFF WHO WRITE PROGRAMS WHICH DEAL WITH THE FINANCIAL RECORDS OR PHYSICAL STOCK RECORDS OF THE COMPANY.
THE DEVELOPMENT AND INCREASING POPULARITY OF SCREEN ORIENTED INTERACTIVE MACHINES LIKE THE HP3000 ALSO HAS SIGNIFICANT RAMIFICATIONS.
THIS HAS CREATED A NUMBER OF AREAS OF HIGH RISK

FRAUD

1.PROGRAMMERS COMPUTER EXPERTISE CREATES POSSIBILITIES FOR UNDETECTED FRAUD WITH LOW PERSONAL RISK (EVEN WHEN DETECTED A SIGNIFICANT PERCENTAGE OF COMPUTER FRAUDS ARE NOT REPORTED; MANY COMPANIES TAKE THE VIEW THAT WITH SMALL PROSPECT OF RECOVERING THEIR MONEY IT IS NOT WORTH THE ADDITIONAL ADVERSE PUBLICITY,EMBARASSEMENT AND POSSIBLE EFFECT ON THEIR SHARE PRICE TO START A PROSECUTION).THIS ATTITUDE DOES OF COURSE ENCOURAGE SUCCESSFUL(UNPROSECUTED) COMPUTER CRIMINALS TO ATTEMPT TO REPEAT THEIR CRIME ON ANOTHER COMPANIES COMPUTER.

NEGLIGENCE

2. THE INTERACTIVE NATURE OF THE HP3000 MEANS THAT PROGRAMMING AND TESTING IS TYPICALLY DONE ON-LINE. THIS CAN CREATE SIGNIFICANT

PROBLEMS. IT IS VERY DIFFICULT TO ENSURE THAT full system testing takes place (particularly in small to medium sized companies) because of the screen oriented nature of HP3000 programs.On-line program testing and debugging by the programmer(or group of programmers) who wrote the program (or module) can lead to inadequate system testing taking place (even in those Companies who have a clearly defined test plan). This is because A) It is extremely difficult to ensure that a programmer has carried out the full system testing that was specified B) That if the full system testing was carried out, that it was carried out in a single pass i.e. a programmer who has spent eight hours(or more) testing a program on a heavily loaded system might be unwilling to repeat the testing from the start if he found a small easily modified "bug" which he felt was not in a section of code that was used during the earlier part of the testing!.
C)multi module screen based programs (with a consequent lack of full testing documentation) written to, what often turns out to be very optimistic, deadlines create strong temptations to make up time by reducing the extent of the testing.

Maintenance and "BUGS"

Fully audited systems with comprehensive testing can still cost Companies hundreds of thousands of dollars through the discovery (and abuse)of "bugs" by users, introduction of program "bugs" through maintenance and or program updates carried out without comprehensive retesting .The effect of an undetected "BUG"getting loose on a "live" stock control system with thousands of transactions per day is almost too horrible to contemplate. IT HAS BEEN DONE ! MANY TIMES ! Not of course by the same Company (some of which are no longer available to tell the tale).

OTHER

There are of course many other potential problem areas. Breaches of system security, updating programs to meet legislative changes (hopefuly before being in breach of the legislation) etc.

The purpose of mentioning the foregoing is to highlight some of the problems HP3000 users have to surmount in ensuring that the programs that are released on their systems match both external and internal audit (internal audit of course includes all quality assurance steps carried out in the DP dept) criteria. This is a difficult and often thankless task, not assisted by the fact that QA is often percieved as an an optional cost ("we would love to spend more time/money on further QA but we don't have the time/money to do more"). Many Companies have taken this approach and many of them through good luck,good judgement,tolerant management/auditors or whatever have not experienced major problems as a result of this approach, despite in many cases effectively having system and program changes retrospectively confirmed by the external auditors.

THE FUTURE

DP has grown from an expensive infant available to a very

select highly trained few, to mass cheap availability. As with most things increased use has led to the discovery of the potential for abuse. At a Corporate level the statistical incidence of the abuse is often minimal. However each failure in Corporate controls has the potential to cost hundreds of thousands of dollars (and perhaps Corporate existence).

External auditors, the SEC etc are aware of the problem and are focussing closely on it. The certain consequence of their attention is of course more guidelines and legislation some of which has already been enacted, some of which is now actively in progress and a great deal of which has yet to fall into our laps to cope with.

Auditors (and auditing) will become more sophisticated (but not cheaper) and more mechanised methods of system and program testing will be introduced.

Old programs will have to be modified to cope with this (so that 1. they can be audited in the future 2. at a reasonable cost).
comprehensive, fully documented professional QA will become increasingly important.
program development will have to be in conjunction with the auditors, to ensure compliance with their requirements.
Systems development and testing will have to take computer audit into account.
current and future data protection requirements need to be considered and implemented prior to (not subsequent) to system implementation.

VARIOUS WAYS TO OPTIMISE QA AND AUDITING.

In conclusion the DP dept has to accept that program development (and in particular that on interactive machines) will be subject to greater audit and QA requirements in the future. Those Corporations that make a virtue out of neccesity i.e use QA as a profit centre and already plan for future requirements in consultation with their auditors, are the ones who can face the future with the greatest confidence.

HOW TO GET THE MOST OUT OF YOUR MRP SYSTEM.

BY

MIKE KOSOLCHAROEN.

MANUFACTURING PRODUCTIVITY DIVISION.

HEWLETT PACKARD COMPANY.

INTRODUCTION.

There are many ways to define a good MRP system. The most common method is by measuring the benefits derived from implementing MRP. Different classes of MRP users are defined for different levels of benefits attained. However, it is also important to heed the cost of maintaining and operating MRP. This cost can be very high if the system is not reliable and is problem-prone. In simple terms, a good MRP system should not require complicated preparations prior to each run nor should it require complicated recovery procedures if the system is interrupted. Ideally, one should be able to start MRP at the end of the day or the week without worrying that he/she may be called at night or during the weekend. In addition, the system should be flexible enough to accommodate common user requirements without extensive modifications. The objectives should be to maximize benefits and to minimize cost.

This paper emphasizes ways to optimize a computerized MRP system from the operations point of view. The discussion applies mainly to a regenerative system and covers the areas listed below:

- Data Accuracy.
- Failure Recovery.
- Common Problems.
- Performance.
- Special Applications.
- Interfacing.

Hewlett Packard's Materials Management/3000 (MM/3000) MRP system will be used as a reference system. This MRP system has been refined through years of actual field applications and extensive enhancements. For those of you who already use Hewlett Packard's MRP system you may benefit from many of the suggested applications.

DATA ACCURACY.

The accuracy of the input data is unquestionably the prerequisite to a successful MRP operation. MRP reports are meaningless if the part or inventory information is not correct. Successful MRP users claim to attain a very high inventory accuracy of 98% or over. One good technique to insure inventory accuracy is Cycle Counting. It allows physical verification on inventory without disrupting the entire operation. The Inventory Management module in MM/3000 has a batch job called "CYCLE INV-COUNT" for this purpose.

FAILURE RECOVERY.

To alleviate the burden of supporting MRP, provisions for recovery must be carefully planned. The recovery process must be simple and easy to use. If the system crashes while running MRP (perhaps due to a hardware failure), when the system is recovered, MRP should be restartable from its last processing point so that very little time is lost. Similarly, if MRP aborts, a recovery program should be automatically activated and should attempt to restart MRP from its last successful execution. All these activities should occur without any users' or programmers' involvement. This method is friendlier than the traditional one which requires purging and recreation of some intermediate files before restarting. In MM/3000, the recovery job is specified in the Job Control Language (JCL) file as "!COMMENT $RESTART Recovery Jobname". This job evaluates files used in the step, purges them if necessary, regenerates the JCL and restarts MRP. Execution will resume at the last processed $CHECKPOINT statement. An example of the JCL is shown in Figure 1.

COMMON PROBLEMS.

There are many common problems that the system administrator of MRP may encounter in maintaining the system. Being aware of these potential problems can help you avoid them. Some of the problems and solutions are discussed below:

a. FILE SIZE PROBLEM. The most common reason for MRP systems to abort is an output file that is full. Unfortunately, recovery procedures may not be simple. Typically the output files for that program must be rebuilt and the program has to be restarted. This process usually requires programmer involvement. In MM/3000, MRP tries to assure you that the files are built large enough to accommodate the environment of your next MRP run. It has

memory! MRP remembers the file sizes used in the previous run. Based on this information and the database strip file sizes of the new run, MRP builds the new files. These new files can be built larger by a certain percentage defined by the user in the form of a system-value called "FILE-GROWTH-PCNT" (default value 10%). For the strip files, MRP builds them programmatically after accessing the data bases. MRP will release unused space of the output files after successful execution of each program. In addition, the system provides run time protection in case a file is still full. In such an event, rather than aborting, MRP will expand the file which is full and continue processing. Hence the probability of having a file size problem is extremely small.

b. HIGH ORDER TRUNCATION. Another common cause of MRP failure is high order truncation or integer overflow which can happen even after many successful runs. The quantity fields such as Order-Quantity can easily have high order truncation with the following scenario:

- A large order quantity for the top level part (e.g. 1,000 units).
- A few structures (e.g. 7 levels), with a fair amount of quantity per (e.g. 5).
- Lot-for-lot order policy.

The order quantity for the lowest level part will be:

= order qty (of top level) * qty-per at each level

$$= 1{,}000 * 5^{7} = 78{,}125{,}000 \text{ !}$$

Solutions to this problem depend on the shop practice. If it is not unusual for the facility to accommodate a very large order, then the quantity fields should be enlarged. The standard MM/3000 field length is 7 digits. However, the maximum field size is 19 digits. The customization feature of MM/3000 allows easy modification to the field length. Otherwise, changing the order policy and/or setting a limit on the order quantity for the higher level part should solve this problem.

c. BAD STRUCTURE. Most MRP systems plan and reschedule by structure level. Low Level Code, an attribute in the structure record, is a key parameter in controlling this process. If a structure is bad or

the Low Level Code is wrong, the MRP planning and scheduling process will be affected. This problem is usually fatal and will cause MRP to abort. To avoid this problem, an edit routine should be created to detect and sort out bad structures. MM/3000's MRP sorts and discards bad structures so that the components of the affected structures will not be planned for. In addition, the Parts and Bills of Material module incorporates structure degeneracy checks which provide immediate feedback via the on-line system.

d. BAD INTERFACE INPUT DATA. A flexible MRP system should allow for interfacing with other systems. Additional data can be funneled to MRP in the form of independent files. This extra channel of input may cause problems. MM/3000's MRP edits all fields in the input records from the independent files and the job will not run until all the bad data is corrected.

PERFORMANCE.

MRP run time is always a concern in selecting or designing a MRP system. In general, MRP performance can be affected by various factors. Some of these are:

a. INPUT DATA. Obviously, the size of the input data affects the run time. MRP data can be divided into four types:

- Part data.
- Structure data.
- Supply data.
- Demand data.

From the surface, it may seem that not much can be done to control the size of data. Surprisingly, in many instances, the part and structure data in the data base may be much more than needed. Archiving or removing inactive data from the data bases is recommended. Another approach is to apply various techniques such as Modular Bills, when feasible, to help reduce the data.

b. ORDER POLICY. Order policies and order quantities may have significant impact on the number of suggested orders MRP must generate. A Lot-for-Lot or Fixed order policy with a small order quantity forces MRP to suggest a very large number of orders to cover the unsatisfied demand. On the contrary, Fixed, Order Point or Days of Supply order policies with large order quantities minimize the number of

suggested orders; hence improving the run time. Selecting an order policy is usually based on the need and the nature of the operation. However, being aware of the consequences and implementing corrections can help improve MRP performance.

c. THE PLANNING HORIZON. The shop calendar is another key factor. The longer the horizon, the more planning activities MRP tends to have. Controlling the Master Scheduling horizon will indirectly improve the performance of MRP. Again, this should only be implemented if there is no impact on the basic business practice.

FEATURES AND SPECIAL APPLICATIONS.

There are many MRP features and applications. Four topics are discussed here:

a. Controlling The Reports.
b. Interpreting The Run Statistics.
c. Accommodating Field Changes.
d. Miscellaneous.

a. CONTROLLING THE REPORTS.

Users of MRP systems have different requirements for reports. Report controlling and modification is one common enhancement requested by MRP users. Some of the requests are:

- Multiple copies.
- Filtering the contents of a report.
- Suppressing certain reports.
- Sending reports to different printers.
- Reproducing certain reports (a report that was lost).

The MRP system should be designed with these capabilities. With MM/3000, you can accommodate these requirements easily. The system produces four reports:

- Action report. (Figure 2.)
- No Activity report. (Figure 3.)
- Exception report. (Figure 4.)
- Controller Summary report. (Figure 5.)

The Action report is the largest. There are three ways to filter the report content:

- Flags. Two fields in the part record: EXPLODE-CODE and ACTION-RPT-FLAG (default = "Y" for both) should be considered. There will be no explosion to the lower level if the EXPLODE-CODE is set to "N". If the ACTION-RPT-FLAG is set to "N", that part will not appear on the Action report.

- System Value. A system value called "MRP-ACTN-RPT-OPT" dictates the criteria for printing the report. There are three possible values:

  0 = Print the Action Report for all parts.

1 = Print the Action Report for parts that have either supply (e.g. work orders, purchase orders) or demand (e.g. extra usages, dependent demand).
2 = Print the Action Report only for parts that have MRP suggested actions such as pull-in, push-out, cancel.

- File equations. If you don't want to see reports for certain Controllers, a simple file equation can suppress them. A comprehensive example is provided in Figure 1.

The No Activity Report and the Controller Summary Report can be suppressed by equating the output files to $NULL. Manipulating the output Spool files enables you to control the printer, the output priority and the number of copies. Figure 1. also demonstrates how to implement them.

You can always reproduce MRP reports without having to rerun the entire MRP job. The Action Report, the No Activity Report and the Controller Summary Report can be reproduced by scheduling job "MRP (PART-2)" with the Current Checkpoint set to "92". The Exception Report can also be reproduced using the same job and Current Checkpoint "93".

The following JCL is part of file MRPJCL2J.PUB:

```
!JOB MRP,MGR.MA
!COMMENT $JOBNAME MRP (PART-2)
!COMMENT $RESTART MRPRCVRJ JOB.............(1)
 .
 .
!COMMENT
!COMMENT $CHECKPOINT92.....................(1)
!COMMENT $STEPNAME MRPACTN
!COMMENT
 .
 .
!FBUILD MRP2501D.MRPWORK;FORMAT=241;DISC=500
!FBUILD MRP2502D.MRPWORK;FORMAT=394;DISC=500
!FBUILD MRP2503D.MRPWORK;FORMAT=241;DISC=500
 .
 .
!FILE MRPACTN2;DEV=LP1,8,1.................(2)
!FILE MRPACT33;DEV=LP2,8,2.................(2)
!FILE MRP2503D.MRPWORK=$NULL...............(3)
!FILE MRP2504D.MRPWORK=$NULL...............(4)
 .
 .
!RUN MRP2500P.HP32260
!RESET MRPACTN2............................(5)
!RESET MRPACT33............................(5)
 .
 .
!COMMENT
!COMMENT $CHECKPOINT93
!COMMENT $STEPNAME MRPEXCPT
 .
 .
!FILE MRPEXC02;DEV=LP1,8,1.................(6)
!FILE MRPEXC33;DEV=LP2,8,2.................(6)
 .
 .
!RUN MRP3000P.HP32260
!RESET MRPEXC02............................(5)
!RESET MRPEXC33............................(5)
```

FIGURE 1. JCL MODIFICATION TO CONTROL REPORTS.

Refer to the next page for explanations of (1) through (6).

Following are notes that describe (1) through (6) in Figure 1.

(1). The $RESTART command will activate MRPRCVRJ recovery job if MRP fails. The $CHECKPOINT, with checkpoint number, is assigned for each program in the JCL. MRP will restart from the last executed $CHECKPOINT.

(2). The Spool file naming convention for the Action Reports are as follow:

MRPACTNn for single digit Controller number. The suffix "n" is the Controller number.
MRPACTnn for double digit Controller number. The suffix "nn" is the Controller number.

In this example, the Action Reports for Controller 2 are sent to printer "LP1", with output priority "8" and the number of copies is "1". The Action Reports for Controller 33 are sent to printer "LP2", with output priority "8" and the number of copies are "2".

(3). This file equation is to suppress the No Activity Report.

(4). This file equation is to suppress the Controller Summary Report.

(5). Using the RESET command is strongly recommended. Otherwise, the maximum number of file equation limit may be encountered.

(6). The Spool file naming convention for the Exception Reports is as follows:

MRPEXCnn for single and double digit Controller number. The suffix "nn" is the Controller number.

CONTROLLER 6 — COMPANY NAME — PAGE 4
RUN DATE 08/20/82 — ACTION REPORT

PT11-A (CONTINUED)

--NEED DATE---	--PROJ-AVAIL-	---QTY--	------PARENT------	----DEMAND-----	----SUPPLY-----	-ITEM-	---ACTION---	-START-DATE-
09/27/82	0	10			SUGG	08030005	W/I WINDOW	08/23/82
11/08/82	-10	-10	PT1-A	EXPLO 08020006				
	0	10			SUGG	08030006		10/04/82

PT12-A PICKUP TAILGATE — ORDER POLICY:FIXED — ABC (CODE): A

0 QTY ON HAND	0 QTY IN INSPECTION	LEAD TIME:	25	6 MONTH REQUIREMENT:	60
0 SAFETY STOCK	0 TOTAL AVAILABLE	SAFETY LEAD TIME:	0	UNIT COST:	-.0005
10 ORDER QTY	0 ORDER QTY MULTIPLE	SETUP COST:	-.0001	ORDER POINT:	0
EA UNITS PURCHASED	PART CLASS	% SHRINKAGE:	0	% YIELD:	100

--NEED DATE---	--PROJ-AVAIL-	---QTY--	------PARENT------	----DEMAND-----	----SUPPLY-----	-ITEM-	---ACTION---	-START-DATE-
08/09/82*	-10	-10	PT1-A	EXPLO 08020001				
	-20	-10	PT1-A	EXPLO 08020002				
	-10	10			SUGG	08030007	PAST DUE	07/05/82+
	0	10			SUGG	08030008	PAST DUE	
08/23/82	-10	-10	PT1-A	EXPLO 08020003				
	0	10			SUGG	08030009	W/I WINDOW	07/19/82+
09/06/82	-10	-10	PT1-A	EXPLO 08020004				
	0	10			SUGG	08030010	W/I WINDOW	08/02/82+
09/20/82	-10	-10	PT1-A	EXPLO 08020005				
	0	10			SUGG	08030011	W/I WINDOW	08/16/82+
11/01/82	-10	-10	PT1-A	EXPLO 08020006				
	0	10			SUGG	08030012		09/27/82

PT2-A CHROME REAR BUMPER — ORDER POLICY:FIXED — ABC (CODE): A

0 QTY ON HAND	0 QTY IN INSPECTION	LEAD TIME:	30	6 MONTH REQUIREMENT:	51
0 SAFETY STOCK	0 TOTAL AVAILABLE	SAFETY LEAD TIME:	0	UNIT COST:	-.0005
10 ORDER QTY	0 ORDER QTY MULTIPLE	SETUP COST:	-.0001	ORDER POINT:	0
EA UNITS PURCHASED	PART CLASS	% SHRINKAGE:	0	% YIELD:	100

--NEED DATE---	--PROJ-AVAIL-	---QTY--	------PARENT------	----DEMAND-----	----SUPPLY-----	-ITEM-	---ACTION---	-START-DATE-
08/16/82*	-12	-12	PT-A	EXPLO 08010001				
	-2	10			SUGG	08020007	PAST DUE	07/05/82+
	8	10			SUGG	08020008	PAST DUE	
08/30/82	-4	-12	PT-A	EXPLO 08010002				
	6	10			SUGG	08020009	W/I WINDOW	07/19/82+
09/13/82	-6	-12	PT-A	EXPLO 08010003				
	4	10			SUGG	08020010	W/I WINDOW	08/02/82+
09/27/82	-8	-12	PT-A	EXPLO 08010004				
	2	10			SUGG	08020011	W/I WINDOW	08/16/82+
11/08/82	-1	-3	PT-A	EXPLO 08010005				
	9	10			SUGG	08020012		09/27/82

TR1000-A MEDIUM SIZE TRUCK — ORDER POLICY:LOT FOR LOT — ABC (CODE): A

0 QTY ON HAND	0 QTY IN INSPECTION	LEAD TIME:	5	6 MONTH REQUIREMENT:	0
0 SAFETY STOCK	0 TOTAL AVAILABLE	SAFETY LEAD TIME:	0	UNIT COST:	-.0005
12 ORDER QTY	0 ORDER QTY MULTIPLE	SETUP COST:	-.0001	ORDER POINT:	0
EA UNITS FABRICATED	PART CLASS	% SHRINKAGE:	0	% YIELD:	100

--NEED DATE---	--PROJ-AVAIL-	---QTY--	------PARENT------	----DEMAND-----	----SUPPLY-----	-ITEM-	---ACTION---	-START-DATE-
	48	48			RUN	00000001	FIRM	08/23/82
	96	48			RUN	00000002	FIRM	09/06/82

* = PAST DUE + = PAST START DATE

FIGURE 2. ACTION REPORT.

CONTROLLER 6 — COMPANY NAME — PAGE 1
RUN DATE 08/20/82 — NO ACTIVITY REPORT

PART NUMBER	ABC	PART CLASS	LOW LEVEL CODE	QUANTITY ON HAND	LAST MRP 6-MO-RQMT	STANDARD UNIT COST	INVENTORY VALUE ($)
WR-F	A	F	0	0	0	-.0005	0

FIGURE 3. NO ACTIVITY REPORT.

RUN DATE 08/20/82 EXCEPTION REPORT FOR ORDER ACTION

PT1-A PICKUP BED ASSY UNIT COST: -.0005
UNIT OF MEASURE: EA ABC CODE: A LEAD TIME: 5 ORDER POLICY: FIXED PART CLASS: FABRICATED

--ORDER NBR	----ITEM	-----VENDOR	----QUANTITY	---START DATE	--NEED DATE	--CUR DUE DATE	-CHG DAYS	-STATUS	--EXCEPT	---ACTION------
06020001			10	08/09/82+	08/16/82*			SUGG	0	PAST DUE
06020002			10	08/09/82+	08/16/82*			SUGG	0	PAST DUE
06020003			10	08/23/82	08/30/82			SUGG	0	W/I WINDOW
06020004			10	09/06/82	09/13/82			SUGG	0	W/I WINDOW

PT11-A PICKUP BED UNIT COST: -.0005
UNIT OF MEASURE: EA ABC CODE: A LEAD TIME: 25 ORDER POLICY: FIXED PART CLASS: PURCHASED

--ORDER NBR	----ITEM	-----VENDOR	----QUANTITY	---START DATE	--NEED DATE	--CUR DUE DATE	-CHG DAYS	-STATUS	--EXCEPT	---ACTION------
06030001	0		10	07/12/82+	08/16/82*			SUGG	0	PAST DUE
06030002	0		10	07/12/82+	08/16/82*			SUGG	0	PAST DUE
06030003	0		10	07/26/82+	08/30/82			SUGG	0	W/I WINDOW
06030004	0		10	08/09/82+	09/13/82			SUGG	0	W/I WINDOW
06030005	0		10	08/23/82	09/27/82			SUGG	0	W/I WINDOW

PT12-A PICKUP TAILGATE UNIT COST: -.0005
UNIT OF MEASURE: EA ABC CODE: A LEAD TIME: 25 ORDER POLICY: FIXED PART CLASS: PURCHASED

--ORDER NBR	----ITEM	-----VENDOR	----QUANTITY	---START DATE	--NEED DATE	--CUR DUE DATE	-CHG DAYS	-STATUS	--EXCEPT	---ACTION------
06030007	0		10	07/05/82+	08/09/82*			SUGG	0	PAST DUE
06030008	0		10	07/05/82+	08/09/82*			SUGG	0	PAST DUE
06030009	0		10	07/19/82+	08/23/82			SUGG	0	W/I WINDOW
06030010	0		10	08/02/82+	09/06/82			SUGG	0	W/I WINDOW
06030011	0		10	08/16/82+	09/20/82			SUGG	0	W/I WINDOW

PT2-A CHROME REAR BUMPER UNIT COST: -.0005
UNIT OF MEASURE: EA ABC CODE: A LEAD TIME: 30 ORDER POLICY: FIXED PART CLASS: PURCHASED

--ORDER NBR	----ITEM	-----VENDOR	----QUANTITY	---START DATE	--NEED DATE	--CUR DUE DATE	-CHG DAYS	-STATUS	--EXCEPT	---ACTION------
06020007	0		10	07/05/82+	08/16/82*			SUGG	0	PAST DUE
06020008	0		10	07/05/82+	08/16/82*			SUGG	0	PAST DUE
06020009	0		10	07/19/82+	08/30/82			SUGG	0	W/I WINDOW
06020010	0		10	08/02/82+	09/13/82			SUGG	0	W/I WINDOW
06020011	0		10	08/16/82+	09/27/82			SUGG	0	W/I WINDOW

TR1000-A MEDIUM SIZE TRUCK UNIT COST: -.0005
UNIT OF MEASURE: EA ABC CODE: A LEAD TIME: 5 ORDER POLICY: LOT FOR LOT PART CLASS: FABRICATED

--ORDER NBR	----ITEM	-----VENDOR	----QUANTITY	---START DATE	--NEED DATE	--CUR DUE DATE	-CHG DAYS	-STATUS	--EXCEPT	---ACTION------
00000001			48	08/16/82+	08/23/82			OPEN	0	FIRM
00000002			48	08/30/82	09/06/82			OPEN	0	FIRM
00000003			48	09/13/82	09/20/82			OPEN	0	FIRM
00000004			48	09/27/82	10/04/82			OPEN	0	FIRM
00000005			12	11/08/82	11/15/82			OPEN	0	FIRM

****** LEGEND : '*' = PAST DUE '+' = PAST START DATE ******

FIGURE 4. EXCEPTION REPORT.

RUN DATE 08/20/82 COMPANY NAME PAGE 1

MRP CONTROLLER SUMMARY REPORT

CTLR	# OF PARTS	TOTAL INVENTORY $	TOTAL SIX MO RQMT $	---NO-ACTIVITY-- PARTS	INV $	----------SUGGESTED-ACTIONS---------- PULL	PUSH	CNCL	NEW
1	24	0	0	0	0	0	0	0	53
2	18	0	-2	1	0	0	0	0	51
3	8	0	0	1	0	0	0	0	48
4	10	0	0	1	0	0	0	0	68
5	11	0	0	1	0	0	0	0	45
6	10	0	0	1	0	0	0	0	52
TOTALS	82	0	-4	5	0	0	0	0	318

FIGURE 5. CONTROLLER SUMMARY REPORT.

b. INTERPRETING THE RUN STATISITCS.

MRP produces run statistics. This information is extremely useful as a problem solving tool or for report integrity checks. You should familiarize yourself with the report statistics when starting up your new MRP system. Some of the Statistics from MM/3000's MRP are discussed below:

Figure 6. shows how many records are stripped from the data bases and which output files they go to.

Figure 7. shows how this data is split into different levels.

Figure 8. shows how each main planning and scheduling program treats the data at each level.

```
HP32260A.06.00    FRI, AUG 20, 1982,  4:22 PM
-------------------------
MRP 1000 EDIT ERROR REPORT
-------------------------
PART NUMBER          ORDER #   RECORD TYPE   ACTION   TYPE   STATUS   GEN-LL-RQMT-FLAG

-----------------------------------------------
     MRP1000        STATISTICS
-----------------------------------------------

SHOP CALENDAR CONTROL DATES

 RUN DATE                    820820    166
 SIX MONTHS DATE             830204    286
 CALENDAR END DATE           831230    521

HIGHEST STRUCTURE LEVEL          5

INPUT DATA SET      RECORDS STRIPPED   OUTPUT FILE

 ITEM DATA                      83      PART
 INV MASTER                     83      PART
 STRUCTURE                      86      STRU
 RUNS                           12      SPLY
 CONTROLLER                      6      CONTROLLER
 ALLOCATION                    100      RQMT

 ELAPSED MINUTES                 2
 CPU SECONDS                    30
-----------------------------------------------
```

FIGURE 6. STATISTICS FROM THE DATABASE STRIP PROGRAM.

```
                                   FRI, AUG 20, 1982,  4:26 PM
-----------------------------------------------------------------
     MRP1100        STATISTICS
-----------------------------------------------------------------

                                  PART          STRUCTURE
                                  ----          ---------
INPUT                              83              86

OUTPUT                             83              86

                                SUPPLY        REQUIREMENT
                                ------        -----------
INPUT                           12                100

OUTPUT LEVEL 00                 12                  0
       LEVEL 01                  0                100
                                -------       -----------
       TOTAL                    12                100

ELAPSED MINUTES                  1
CPU SECONDS                      7
-----------------------------------------------------------------


-----------------------------------------------------------------
     MRP1200        STATISTICS
-----------------------------------------------------------------

                                  PART          STRUCTURE
                                  ----          ---------
INPUT                              83              86

OUTPUT LEVEL 00                    7               2
       LEVEL 01                   14              32
       LEVEL 02                   29              28
       LEVEL 03                   27               4
       LEVEL 04                    4               2
       LEVEL 05                    2               0
       NOT USED                    0-             18-
                                -------       -----------
       TOTAL                      83              68

ELAPSED MINUTES                  1
CPU SECONDS                      8
-----------------------------------------------------------------
```

FIGURE 7. STATISTICS FROM THE SPLIT PROGRAMS.

```
  HP32200A.00.00    FRI, AUG 20, 1982,  4:29 PM
-------------------------------------------------------------
     MRP2000    STATISTICS FOR LEVEL  0
-------------------------------------------------------------

PART RECORDS                          7
STRUCTURE RECORDS                     2
SUPPLY RECORDS                       12

ORDER POLICY COUNTS
    LOT FOR LOT                       7

REQUIREMENT COUNTS          GENERATED        TOTAL
    LEVEL 01                       12          112
                              -------     -------
    TOTAL                          12          112

ACTION REPORT RECORDS                12

DATABASE UPDATES:
  ITEM-DATA                           7

LARGEST PLAN TABLE SIZE               7
PLAN TABLE LIMIT                    587

ELAPSED MINUTES                       1
CPU SECONDS                           8
-------------------------------------------------------------

  HP32200A.00.00    FRI, AUG 20, 1982,  4:32 PM
-------------------------------------------------------------
     MRP2000    STATISTICS FOR LEVEL  2
-------------------------------------------------------------

PART RECORDS                         29
STRUCTURE RECORDS                    28
REQUIREMENT RECORDS                 103

ORDER POLICY COUNTS
    FIXED                            11
    LOT FOR LOT                      18

REQUIREMENT COUNTS          GENERATED        TOTAL
    LEVEL 01                        0          112
    LEVEL 02                        0          103
    LEVEL 03                      144          144
    LEVEL 04                        6            6
                              -------     -------
    TOTAL                         150          385

ACTION REPORT RECORDS               206

DATABASE UPDATES:
  ITEM-DATA                          29

LARGEST PLAN TABLE SIZE               6
PLAN TABLE LIMIT                    587

ELAPSED MINUTES                       1
CPU SECONDS                          18
-------------------------------------------------------------
```

FIGURE 8. STATISTICS FROM THE PLANNING PROGRAMS, BY LEVEL.

have reservations about the run time of Master Production Scheduling. With MM/3000's new Master Production Scheduling feature called "Fast Final", manufacturing schedules can be created quickly.

If additional input information is desired, independent input files can be added. MM/3000's MRP has two optional input files: the Independent Supply file (MRP1009D.MRPWORK) and the Independent Requirement file (MRP1008D.MRPWORK). You can add manufacturing orders, purchase orders and work orders to the supply file; extra usages, backorders and allocations to the requirement file. This information will be processed along with the data stripped from the databases.

b. OUTPUT.

The outputs of MRP are usually reports. However, many MRP systems generate update files for updating other systems such as the Inventory or Parts Management system. MM/3000's MRP produces four update files for automatic updating of these two systems. Four update jobs are available:

- "UPDATE ABC" to update ABC codes and six month requirements. Input file is MRP2101D.MRPWORK.
- "UPDATE WO DATES" to update the new due dates of work orders. Input file is MRP2102D.MRPWORK.
- "UPDATE EC DATES" to update the engineering changes. Input file is MRP2103D.MRPWORK.
- "UPDATE PO DATES" to update the new due dates of purchase orders. Input file is MRP2104D.MRPWORK.

There are basically two ways to implement MRP'suggestions: manually or programatically. A good selection criteria is by evaluating the report. If you want to implement most MRP suggestions, use the batch job. Remove the update records that you don't want to implement via the EDITOR. Otherwise, update the databases manually through the transactions, if you follow only a few MRP suggestions. MRP output can also be used for Capacity Requirements Planning. There is a job which strips file MRP2402D.MRPWORK and passes the information to Production Management/3000.

SUMMARY.

To summarize, there are many operational considerations that can be addressed to help optimize your MRP system. You may want to look at the following areas:

1. Is your data accurate. Can you use cycle counting?

2. Are you familiar with the recovery process on your system and is it automatic?

3. Do you share the common problems with other MRP users and can you correct them?

4. Can you improve the run time by:

   - Using modular bills where practical.
   - Adjusting order quantities and order policies.
   - Controlling the Master Schedule horizon.

5. Do you take advantage of your system's special features?

   - Are you controlling your reports so you receive only the data you need?
   - Do you use the run statistics as a problem solving tool?

6. Finally, are you making use of the ability to interface your MRP system with other planning and control modules such as Master Scheduling, Work Order and Purchase Order Control, Inventory Control and Capacity Requirements Planning?

Performance Testing of Five Languages

by
Jim Kramer
HP San Diego

I. Introduction

I have timed various constructs of five of the programming languages on the HP 3000, and this is a report on the results. The five languages are BASIC, COBOL, Fortran, Pascal, and SPL. For BASIC I tested both the compiler and the interpreter, and I tested both COBOL compilers -- COBOL '68 and COBOL '74.

Both CPU and wall time were measured, but I only report here on CPU time. This is because wall time is an issue only in those few tests which involve disc I/O. These include four sequential file access tests and a test of the BASIC interpreter INVOKE statement. For the file I/O tests I mininized disc I/O and wall time by choosing the largest possible blocking factors.

A few years ago when the Series III was the largest 3000, minimizing CPU usage was not a major concern. Because of the nature of business processing and of the Series III, the limiting resource on most machines was disc accesses (this was less true on Series 30's and 33's). However CPU usage is becoming more and more of an issue since the arrival of the Series 40, 44, and 64, even though the CPU power of these machines is much greater than the earlier machines. This is because the potential for I/O concurrency has increased even more than the CPU increase, and because more and more users are using office automation products. Word processing and graphics can be very CPU intensive.

I hope that this study will be a help in optimizing CPU usage on the 3000, and serve as a start toward answering such frequently asked questions as:

--Is COBOL '74 faster than COBOL '68?

--Do Pascal, Fortran, and (compiled) BASIC come close to SPL for execution efficiency?

--How good is COBOL for computational tasks?

--Can the I/O constructs in the languages come close to the speed of calling FREAD and FWRITE directly? And how fast are FREAD and FWRITE?

--How much speed can be gained by compiling a BASIC program?

II. Test Selection

There are many more language constructs than I could reasonably test. Generally I tested those features which were commonly used and available in more than one language. Thus I tested the move of an integer, but not the union of sets in Pascal.

An exception was some COBOL data types not available in the other languages, namely packed decimal and display (ASCII) numbers. Because of the dominant use of COBOL on the 3000, I ran tests on various operations on these data types.

Because of the limited nature of the tests one should try to avoid jumping to conclusions. For instance the tests show BASIC I/O to be slower than any other language, but only one form of I/O was tested: READ/PRINT of a binary file. This leaves untested such constructs as LINPUT, I/O to an ASCII file, and I/O to a formatted file.

III. The Timing Method

The timing tools on the 3000 are the intrinsics TIMER and PROCTIME. TIMER measures wall time and PROCTIME measures CPU time used by the calling process. The resolution of these intrinsics is one millisecond, which is far longer than the execution of most of the constructs being measured. Therefore it was necessary to time many executions of the construct and divide the resulting time by the number of executions to arrive at the time for a single execution.

The problem with this is that the code required to repeatedly execute the construct -- the loop -- also consumes time. Therefore I timed two loops, one enclosing the construct being measured, and one enclosing nothing, and assumed that the difference was due to the construct being measured.

Another problem is that a language construct can have many different execution times depending on the context in which it appears. The context can affect the timing in two ways: a compiler may generate different machine code for apparently identical constructs in different circumstances, and the speed at which the machine executes code may depend on what code executed previously. An example of the former is the COBOL '68 compiler, which may generate as few as two or as many as seven words of machine instructions for a move of a simple integer (PIC S9(4) COMP) variable. The two word code generation depends on the variables being located in the address range of DB+0 to DB+255. An example of a machine executing identical code at differents rates is the Series 64 which can execute much more rapidly if the required code and data are in the cache. (Because of this effect, it is probably better not to run tests of the languages on the 64. Such tests should also not be used to compare the 64 with other machines, because the locality of the tests resulting from the looping makes the cache hit rate unusually high.)

Another question is the accuracy of the TIMER and PROCTIME intrinsics. Certainly TIMER is closely related to the timing mechanism which maintains time of day, and therefore should be expected to be reasonably accurate. PROCTIME is another matter. In a multiprogramming environment a process is continually being interrupted by I/O interrupts, and losing the processor to higher priority processes. It would be surprising if PROCTIME were highly accurate in such an environment, and in fact I was only able to get consistent results running my test job stand alone. I also ran it in a priority Queue with a large quantum to minimize priority re-evaluation.

To maintain as much consistency across languages as possible and to ease the programming task as much as possible, I used a single set of SPL procedures to take the times and record results. All the tests had the following form:

```
CALL STARTLOOP

  Empty Loop

CALL ENDLOOP
CALL STARTTEST

  Loop on Test Construct

CALL ENDTEST
```

The procedures STARTLOOP and ENDLOOP measure the execution time of a null loop, and the procedures STARTTEST and ENDTEST measure the execution time of the loop with test

code. ENDTEST also logs the results to a log file. This technique is not without its own flaws: as we shall see, different languages take different amounts of time to call SPL procedures. However if the loop is repeated enough, this effect should be minimized. I arranged that all of the tests (with a very few necessary exceptions) would execute for more than a second. Thus the calling differences, measured in microseconds, are negligible.

All testing was done on the Series 44 demo system in the San Diego sales office. The software versions were:

MPE	C.D0.20	(Ciper)
BASIC	B.00.18	
BASICOMP	B.00.18	
COBOL '68	C.02.10	
COBOL '74	A.00.08	
FORTRAN	B.01.08	
PASCAL	A.00.03	
SPL	A.08.01	

The test programs have been put on the swap tape for these meetings.

IV. The Results

The results are presented in detail in the table at the end of this paper. The numbers are CPU usage in microseconds (or milliseconds if followed by an asterisk) rounded to two digits. Repeated stand alone runs showed a variation of at most one half of one per cent from run to run. I obtained the numbers shown by averaging six stand alone runs.

First a brief summary of the results.

SPL, Fortran, Pascal and BASIC are very close to equal with the following exceptions: Fortran I/O and BASIC string moves are much slower than the others, and BASIC I/O is much, much slower.

The two COBOL's were much, much slower than these others for computation and data movement, but just as good for I/O. To my surprise, COBOL '68 was faster than COBOL '74 for most operations, and sometimes much faster. A notable exception here was the PERFORM statement, where COBOL '74 is about six times as fast.

The BASIC interpreter is the slowpoke in almost all cases, which is not a surprise nor even a criticism.

Now lets look at the results in more detail.

Data moves were covered by tests 1 through 17. We see in the very first test that COBOL is slower than the other languages for even the simplest operations. This reflects the work COBOL goes through to access its data. The other languages access simple variables with a single instruction. I decided to take a detailed look at what COBOL was doing.

Both COBOL '68 and COBOL '74 keep pointers to their data in code and begin an access by loading a pointer; COBOL '68 loads to top of stack and COBOL '74 loads to the index register. Both then load data to top of stack with the second instruction, but COBOL '68 must spend a third instruction deleting the address from top of stack. The store sequences are similar, but COBOL '68 must execute a fourth instruction interchanging value and address at the top of stack. Thus COBOL '68 executes 7 instructions and COBOL '74 only 4. In spite of that the execution times are almost identical, because in its data accesss COBOL '74 addresses indirectly through a base address in memory. COBOL '74 is actually slower on its double integer move because it loads and stores the two words separately, whereas COBOL '68 treats them as a unit with the double word load and store instructions.

For COBOL an unsigned integer move (PIC 9(4) COMP) takes more work than a signed integer move, because COBOL takes the absolute value of the source before storing. Thus you should work with signed integers unless you really want this operation to take place.

I investigated why COBOL '68 took so much longer than COBOL '74 to move packed numbers (tests 6 and 7). COBOL '74 is just using the SLD (shift left decimal) instruction to move the source to destination, whereas COBOL '68 moves the source to an intermediate location with SLD, uses another SLD in place at the intermediate location, and then moves to the destination with a MVB (move bytes). Obviously no optimization has been done for the simple special case.

The string moves suggest that Pascal may be the language of choice for convenient and efficient string handling. Both Pascal and BASIC offer the very convenient variable length string data type, but Pascal moves a string almost as quickly as FORTRAN and SPL move their fixed length strings whereas BASIC takes an extra 120 microseconds or so. Of course there are many other aspects of string handling that were not tested.

All of the arithmetic tests (18 through 93) perform an operation on two different variables of the same type, and store the result in a third variable of the same type. COBOL has many ways to specify arithmetic operations; for example the following three statements all specify the same

operation:

```
ADD A TO B.
ADD A , B GIVING B.
COMPUTE B = A + B.
```

I chose to test two of these forms for COBOL, the GIVING form and the COMPUTE form. For COBOL '74 there was no difference in execution speed between the two forms. For COBOL '68 the COMPUTE form was somewhat slower.

The most notable aspect of these results is the speed advantage that COBOL '68 has over COBOL '74. The reason is that COBOL '74 converts its operands to a type with higher precision before doing the computation. Integers are converted to double integers, and double integers are converted to packed decimal. The results suggest that it is probably wise to avoid the use of double integers in COBOL '74 (PIC S9(5) to S9(9) COMP) since packed decimal is faster.

Not surprisingly, arithmetic on DISPLAY numbers is very slow, because they must be converted to a type on which the 3000 hardware can do arithmetic (presumably packed decimal).

In view of the above the results of the double precision divide are a great surprise: COBOL '74 is enormously faster than COBOL '68. For this operation the two compilers have changed philosophies. COBOL '74 performs no conversions on the operands and uses the hardware double integer divide instruction (DDIV). COBOL '68 converts both operands to packed decimal and calls a procedure (DIVD) to perform the divide, since there is no decimal divide instruction.

Some of the loop control constructs are tested in 94 through 104. In all cases the timings are for one time through the loop. The timing methods were altered slightly for these tests. There was no code between the calls to STARTLOOP and ENDLOOP, and the loop being tested was put between the calls to STARTTEST and ENDTEST. In the case of the COBOL PERFORM's, the paragraph being PERFORM'ed was null.

Most notable is the tremendous speed of the COBOL '74 PERFORM compared to that of COBOL '68. This is no doubt due to the new instructions added to the 3000 to support COBOL '74.

Subroutine (and for COBOL, paragraph) calling mechanisms were tested in tests 105 through 118. In all cases the code being called had an immediate return. In the calls with four parameters, the parameters were two integer arrays of 10 words each, and two integers. In the cases where Fortran, Pascal or SPL procedures were called, the called procedures were placed in a different segment from the

calling code.

The speed of the COBOL PERFORM is again noteworthy, as is the speed of the INVOKE in compiled BASIC (it is implemented with the PCAL instruction, which is the basic procedure call mechanism on the 3000). The INVOKE in interpreted BASIC is very slow because of some data swapping to disc. However this mechanism allows the interpreter to run some programs which cannot run compiled because of stack size limitations.

Calls to COBOL subprograms are slow because of initialization code in the subprograms themselves. This is worse for dynamic subprograms, but it should be mentioned that dynamic subprograms are usually preferable anyway because of stack savings.

Only the simplest form of disc I/O was tested: sequential access to a fixed record file (tests 123-126). The very same files were used for all languages, and they were made binary to test BASIC's binary READ and PRINT (which operate differently than ASCII READ and PRINT). The blocks were made as large as possible, as should usually be done for sequential I/O to minimize disc accesses. The 80 byte records were blocked 255, and the 256 byte records were blocked 109 (MPE now allow blocks to be as large as 28000 bytes).

SPL called the FREAD and FWRITE intrinsics directly. The verbs READ and WRITE were used for COBOL, Pascal and FORTRAN.

The results show all the languages to be very close except for BASIC and FORTRAN, which are much slower. The FORTRAN transfers were binary (unformatted) transfers of integer arrays (no loop in the I/O statement), and I suspect that this is the fastest FORTRAN I/O available. The BASIC READ's and WRITE's contained a loop (FOR I=1 TO N,A(I)), and I suspect that this is the cause of the very slow execution -- note the great increase in time to transfer 256 bytes over 80 bytes.

Test	Description	BASIC Comp.	BASIC Int.	COBOL '68	COBOL '74	FORT	PASCAL	SPL
1	Integer Move	2.7	370	9.2	8.8	2.8	2.7	2.7
2	Unsigned Integer Move			11	10			2.7
3	Double Integer Move			12	16	4.5	4.5	4.5
4	Real Move	4.5	370			4.5	4.5	4.5
5	Long Real Move	12	370			13	9.8	9.0
6	PIC S9(7) COMP-3 Move			80	41			
7	PIC S9(15) COMP-3 Move			90	45			
8	Move 20 Bytes	150	510	20	28	24		24
9	Move 20 Bytes -- Unpacked CHAR Array						24	
10	Move 20 Bytes -- Packed CHAR Array						24	
11	Move 20 Bytes -- String						29	
12	Move 80 Bytes	190	580	57	65	61		61
13	Move 80 Bytes -- Unpacked CHAR Array						63	
14	Move 80 Bytes -- Packed CHAR Array						61	
15	Move 80 Bytes -- String						66	
16	Move 10 Words						21	20
17	Move 40 Words						58	57
18	Integer Add	4.4	580			4.2	4.3	4.1
19	Integer Add Giving			12	24			
20	Integer Add (Compute)			12	24			
21	Unsigned Integer Add							4.8
22	Unsigned Integer Add Giving			14	26			
23	Unsigned Integer Add (Compute)			14	26			
24	Double Integer Add					7.1	8.5	7.1
25	Double Integer Add Giving			19	230			
26	Double Integer Add (Compute)			19	230			
27	Real Add	12	590			11	11	11
28	Long Real Add	24	600			24	21	21
29	PIC S9(7) COMP-3 Add Giving			130	160			
30	PIC S9(7) COMP-3 Add (Compute)			160	160			
31	PIC S9(15) COMP-3 Add Giving			150	180			
32	PIC S9(15) COMP-3 Add (Compute)			170	180			

CPU Timings for Various Language Constructs
(Test Times Are in Microseconds
Except * Denotes Milliseconds)

Test	Description	BASIC Comp.	BASIC Int.	COBOL '68	COBOL '74	FORT	PASCAL	SPL
33	PIC S9(7) DISPLAY Add Giving			190	230			
34	PIC S9(7) DISPLAY Add (Compute)			240	230			
35	PIC S9(15) DISPLAY Add Giving			260	310			
36	PIC S9(15) DISPLAY Add (Compute)			300	310			
37	Integer Subtract	4.4	580			4.2	5.6	4.2
38	Integer Subtract Giving			11	24			
39	Integer Subtract (Compute)			12	24			
40	Unsigned Integer Subtract							5.1
41	Unsigned Integer Subtract Giving			14	26			
42	Unsigned Integer Subtract (Compute)			14	26			
43	Double Integer Subtract					7.1	7.2	7.1
44	Double Integer Subtract Giving			19	230			
45	Double Integer Subtract (Compute)			20	230			
46	Real Subtract	12	590			12	11	12
47	Long Real Subtract	24	600			24	21	21
48	PIC S9(7) COMP-3 Subtract Giving			130	160			
49	PIC S9(7) COMP-3 Subtract (Compute)			160	160			
50	PIC S9(15) COMP-3 Subtract Giving			150	180			
51	PIC S9(15) COMP-3 Subtract (Compute)			170	180			
52	PIC S9(7) DISPLAY Subtract Giving			190	230			
53	PIC S9(7) DISPLAY Subtract (Compute)			240	230			
54	PIC S9(15) DISPLAY Subtract Giving			260	310			
55	PIC S9(15) DISPLAY Subtract (Compute)			300	310			
56	Integer Multiply	7.3	580			7.2	43	7.2
57	Integer Multiply Giving			38	27			
58	Integer Multiply (Compute)			38	27			
59	Unsigned Integer Multiply							7.7
60	Unsigned Integer Multiply Giving			40	30			
61	Unsigned Integer Multiply (Compute)			40	30			
62	Double Integer Multiply					11	11	11
63	Double Integer Multiply Giving			220	300			
64	Double Integer Multiply (Compute)			220	300			

CPU Timings for Various Language Constructs
(Test Times Are in Microseconds
Except * Denotes Milliseconds)

Test	Description	BASIC Comp.	BASIC Int.	COBOL '68	COBOL '74	FORT	PASCAL	SPL
65	Real Multiply	16	590			16	16	16
66	Long Real Multiply	32	610			33	30	30
67	PIC S9(7) COMP-3 Multiply Giving			190	220			
68	PIC S9(7) COMP-3 Multiply (Compute)			220	220			
69	PIC S9(15) COMP-3 Multiply Giving			220	260			
70	PIC S9(15) COMP-3 Multiply (Compute)			220	260			
71	PIC S9(7) DISPLAY Multiply Giving			260	300			
72	PIC S9(7) DISPLAY Multiply (Compute)			290	300			
73	PIC S9(15) DISPLAY Multiply Giving			350	410			
74	PIC S9(15) DISPLAY Multiply (Compute)			350	410			
75	Integer Divide	8.6	590			8.8	8.6	8.6
76	Integer Divide Giving			15	18			
77	Integer Divide (Compute)			15	18			
78	Unsigned Integer Divide							8.6
79	Unsigned Integer Divide Giving			17	20			
80	Unsigned Integer Divide (Compute)			17	20			
81	Double Integer Divide					12	12	12
82	Double Integer Divide Giving			490	30			
83	Double Integer Divide (Compute)			550	30			
84	Real Divide	19	600			19	19	19
85	Long Real Divide	47	630			47	45	45
86	PIC S9(7) COMP-3 Divide Giving			450	480			
87	PIC S9(7) COMP-3 Divide (Compute)			550	480			
88	PIC S9(15) COMP-3 Divide Giving			530	560			
89	PIC S9(15) COMP-3 Divide (Compute)			580	560			
90	PIC S9(7) DISPLAY Divide Giving			510	560			
91	PIC S9(7) DISPLAY Divide (Compute)			620	560			
92	PIC S9(15) DISPLAY Divide Giving			640	690			
93	PIC S9(15) DISPLAY Divide (Compute)			710	690			
94	FOR Loop, Integer Loop Index	2.8	270				2.8	2.8
95	DO Loop, Integer Loop Index					2.8		
96	FOR Loop, Double Integer Loop Index						15	

CPU Timings for Various Language Constructs
(Test Times Are in Microseconds
Except * Denotes Milliseconds)

Test	Description	BASIC Comp.	BASIC Int.	COBOL '68	COBOL '74	FORT	PASCAL	SPL
97	DO Loop, Double Integer Loop Index					17		
98	FOR Loop, Real Loop Index	21	280					
99	WHILE Loop, Integer Loop Index						8.6	6.8
100	WHILE Loop, Double Integer Loop Index						15	13
101	REPEAT Loop, Integer Loop Index						9.2	
102	REPEAT Loop, Double Integer Loop Index						13	
103	PERFORM N TIMES Loop			110	18			
104	PERFORM THROUGH N TIMES Loop			110	18			
105	Call to SPL PROCEDURE, No Parameters	31	2.0*	17	21	17	27	17
106	Call to SPL PROCEDURE, Four Parameters	40	2.4*	25	36	25	34	23
107	Call to SPL SUBROUTINE, No Parameters							4.2
108	Call to SPL SUBROUTINE, Four Parameters							11
109	Call to PASCAL PROCEDURE, No Parameters						17	
110	Call to PASCAL PROCEDURE, Four Params						23	
111	Call To COBOL SUBPROGRAM, No Parameters			75	100			
112	Call To COBOL SUBPROGRAM, Four Parameter			110	170			
113	Call To COBOL DYNAMIC SUB, No Parameters			220	190			
114	Call To COBOL DYNAMIC SUB, Four Params			270	260			
115	Call To FORTRAN SUB, No Parameters					19		
116	Call To FORTRAN SUB, Four Parameters					31		
117	PERFORM A Paragraph			99	16			
118	PERFORM THROUGH Paragraphs			99	16			
119	Invoking a BASIC Program	34	95*					
120	GOSUB To a Subroutine	11	110					
121	Call to Multiline Function, One param	22	890					
122	Call to Multiline Function, Four params	27	1.2*					
123	Write Sequential File, 80 Byte Records	7.0*	34*	1.9*	2.0*	4.1*	2.0*	1.8*
124	Read Sequential File, 80 Byte Records	7.1*	39*	1.7*	1.8*	3.9*	1.9*	1.7*
125	Write Sequential File, 256 Byte Records	18*	100*	2.2*	2.2*	4.5*	2.4*	2.1*
126	Read Sequential File, 256 Byte Records	19*	120*	1.9*	2.0*	4.1*	2.2*	1.8*

CPU Timings for Various Language Constructs
(Test Times Are in Microseconds
Except * Denotes Milliseconds)

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 programs 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.

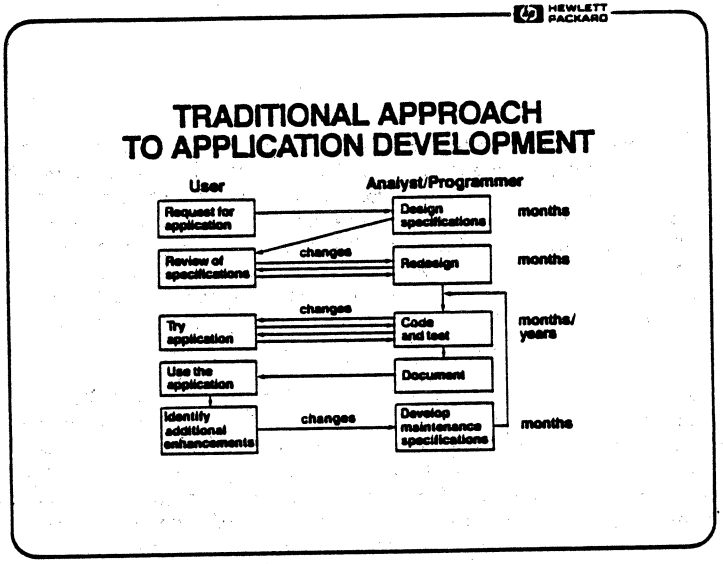
Traditional Approach to Application Development

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.



- 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 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.

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.

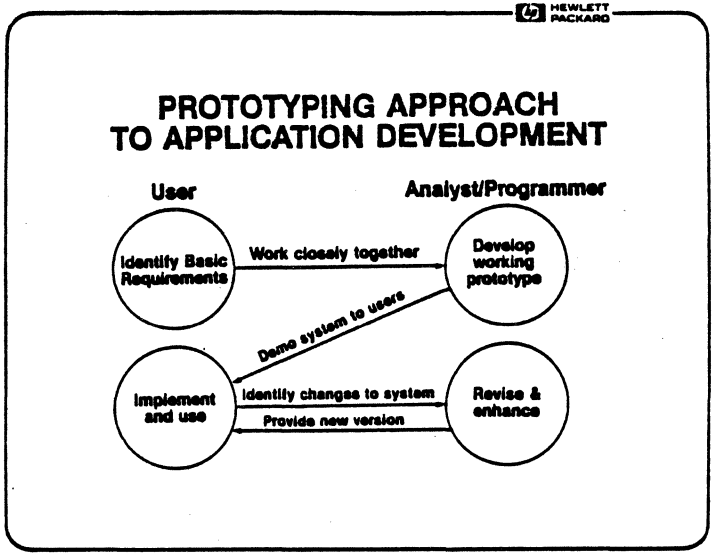
Software Prototypes

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.



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.

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 prototyping:

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 interactive 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 statements. 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 protctyping? 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 prototype. 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 development 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 language, 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 Series 39, 40, 44, 64
  - 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

Bibliography

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 Prototyping 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.

System Optimization at the Programmer Level

Kim Leeper
Wick Hill Associates

This discussion addresses a fundamental best expressed in the cliche: The architecture of the HP3000, as with other machines, is such that inefficiently designed programs directly affect response times throughout the system.

But programming on the HP3000 for maximum efficiency, certainly our common goal, is a skill that must be learned. The majority of us aquire and sharpen this basic skill by practical experience of and coping with performance problems caused by ---- directly related to ---- inefficient programming.

The thrust of this discussion is the providing to the inexperienced programmer points to be considered during new program development, and ways to identify and improve performance of programs already on the system. Among other matters, this discussion will cover:

Comparison of the HP3000 capabilities with several microprocessors, use of the HP3000 as though a microprocessor for the writing of more efficient programs, discussion of the maintaining of major execution loops in a single segment for maximum efficiency, file system considerations, and introduction to tools available for the identification and correction of segmentation problems.

I) Comparisons of HP3000 versus various microprocessor systems:

A) Data and Code Area

1) HP3000

2) IBM-PC

3) Apple IIe

B) Instruction Set Comparison

1) The HP instruction set is more powerful than microprocessor sets.

2) The speed of stack operations on the HP3000 is not necessarily that much faster than microprocessors for simple instructions.

C) Operating Systems

1) MPE

2) CPM

3) MS-DOS

D) Disk Access Time

1) HP Disks

2) Floppies used with microcomputers

3) Winchesters used with microcomputers

E) The Advantages of the HP3000 Over Pure Micro Solutions

1) Shaved data and data bases

2) High level tools to speed development

II) If one designs/implements programs on the HP3000 as though it were a microprocessor system, you will automatically produce more efficient code.

III) Keep your major execution loops in one segment, but remember that the 4-6 kw code segment rule is not written in stone, (i.e. Don't become dogmatic about small code segments). Remember disk access time from microprocessor experience.

IV) File System considerations.

V) Tools to identify segmentation problems.

VI) Conclusion

A Wrong Angle Lense

Ken Lessey
DataCon of St. Helens
50 West Street
St. Helens, Oregon 97051

INTRODUCTION

Photographers are well known for looking at the world in a special way. They use a myriad of different lenses to capture everything from close-up shots to panoramic views. Those of us in data processing view the world in our own special way. We look at every problem in such a way that we can break it down into small pieces of data which can be fed into a computer, digested, and then emerge as the answer to management's questions.

The biggest problem arises because we have only one lense on our camera, and it is set to take close up views. In order to supply a panorama needed by management, we must painstakingly reassemble the data to produce a broad overview. It seems clear that we are approaching this task from the wrong angle. We need to add something to our bag of tricks to allow us to approach this task in a more straight forward manner.

The concept of a profile is similar to that of a relational data base. Using profiles allows us to define alternate ways to look at information. Profiles are primarily used to display information on reports or on inquiry screens. They offer the power and flexibility to process the data before placing it in the profile record. This processing capability makes profiles even more powerful, in this respect, than relational data bases.

THE NEED FOR A PROFILE

Data sets are designed for the HP 3000 with information broken down into its smallest natural groups. Reports, on the other hand, normally require a view of the data that spans several data sets. Reports also require that the data be assembled according to some specific logic.

For example, to produce payroll reports, we need data from five separate areas for complete information about a check. We need:

1. employee demographic data;
2. check net & gross pay data and the date;
3. pay data (current amount plus YTD data);
4. deduction data (current plus YTD);
5. tax data (current plus YTD).

A profile could be used to assemble the data from all these areas so that it can readily be accessed for the reports.

ADVANTAGES

The data in a profile is available for several functions, such as creating a pay register, writing checks, or various reports. When a profile is enhanced or modified, the change affects all functions for which that profile is used. This means that the programmers are not required to "fix" or enhance each individual report. Using profiles allows us to assign experienced programmers to write the profile and then assign entry level people to use the profiles to write reports.

DEFINING A PROFILE

Defining a profile requires that you specifically look at how the data will be reported. Determine, from the user's point of view, what a logical set of data is composed of. You then define how that data is to be selected.

In our example, we selected the data for one check. We may, at some time, also want the data for all checks issued to a specific person during a selected time period. In that case, we are not only combining the data from multiple data sets, but we are also processing that data to compute, for example, gross pay across time.

EXAMPLES

> In an accounts payable system the invoice header along with each line item on the invoice might compose a profile. Data about a specific check and the invoices paid by that check might also compose a profile.

In general, I look for the data that makes up a complete set of information through the eyes of a user. We are actually reassembling the information we broke down when we designed the data base for the system.

In order to correctly define profiles, we must take off the data base designer's hat and put on the user's hat so that we can look at the world from his perspective. That perspective will define the logical sets of data which are most needed by the user. A profile's function is to convert data from the perspective of a data base to data which is useful from the view point of a user.

IMPLEMENTING A PROFILE

Implementing a profile requires that we write a main program which calls the profile. The profile, in turn, assembles data for the profile record which is then passed back to the main program. The main program uses that data to write one line of a report. The main program continues to call the profile and process the data until the function is complete.

The following pages contain a diagram of the part of the data base we have been using as an example, flow charts for the main program and the profile procedure, and pseudocode for the main program and the profile procedure.

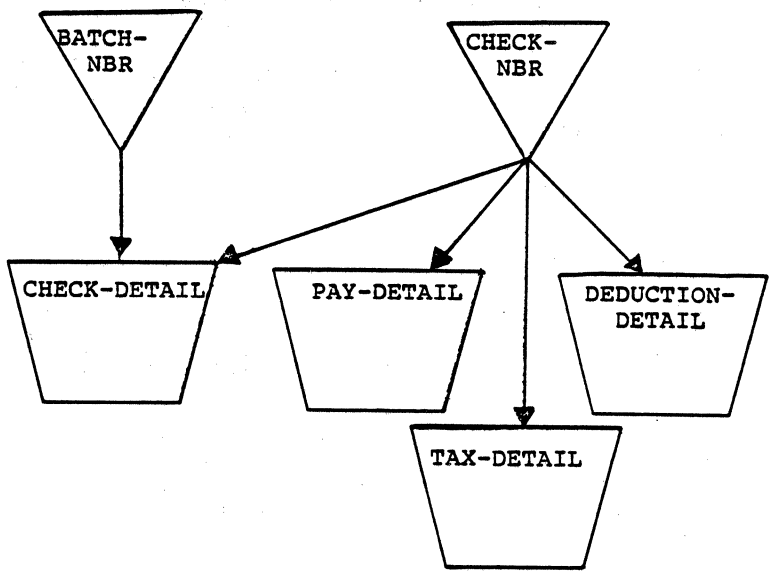
CONCLUSION

We are now using profiles extensively in our shop and have discovered some major benefits. The task involved in generating reports has been reduced from a major problem in logic and reading the data base to a decision on how to format data on a page to be most useful and pleasing to the user.

Because we don't spend time solving complex problems, the cost of our reports has been reduced to approximately 25% of the original cost. We have also noticed that the reliability of these reports has increased significantly. The reports written with profiles are much easier to enhance and maintain.

In short, by using profiles, we write much better reports in less time and for less money. Our clients are as happy about that as we are.

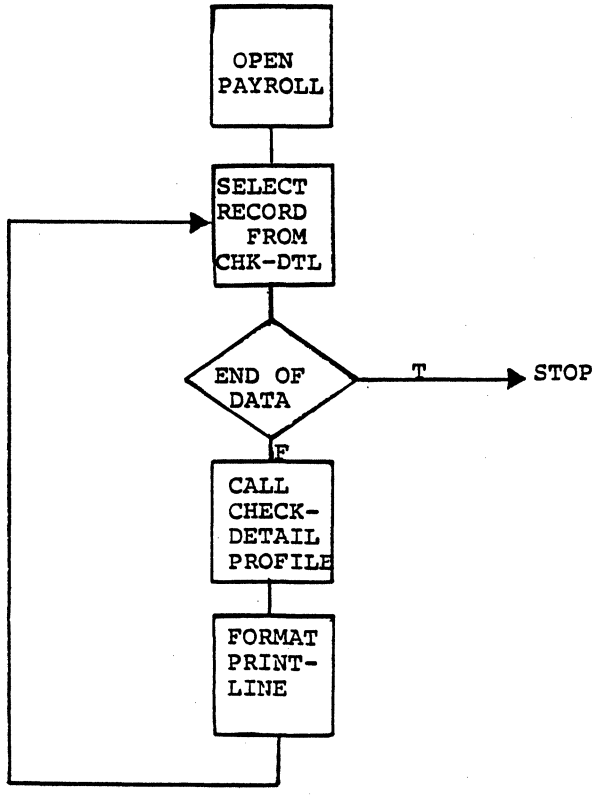
PAYROLL DATA BASE



PSEUDOCODE FOR A MAIN PROGRAM

```
OPEN PAYROLL
REPEAT
   SELECT NEXT RECORD FROM CHECK-DETAIL DATA SET
     WITH BATCH-NBR = SELECTED BATCH-NBR
   IF END-OF-DATA-SET
     EXIT
   ELSE
     CALL CHECK-DETAIL (name of profile)
     PERFORM FORMAT-PRINT-LINES
   ENDIF
ENDREPEAT
```

FLOW CHART - MAIN PROGRAM



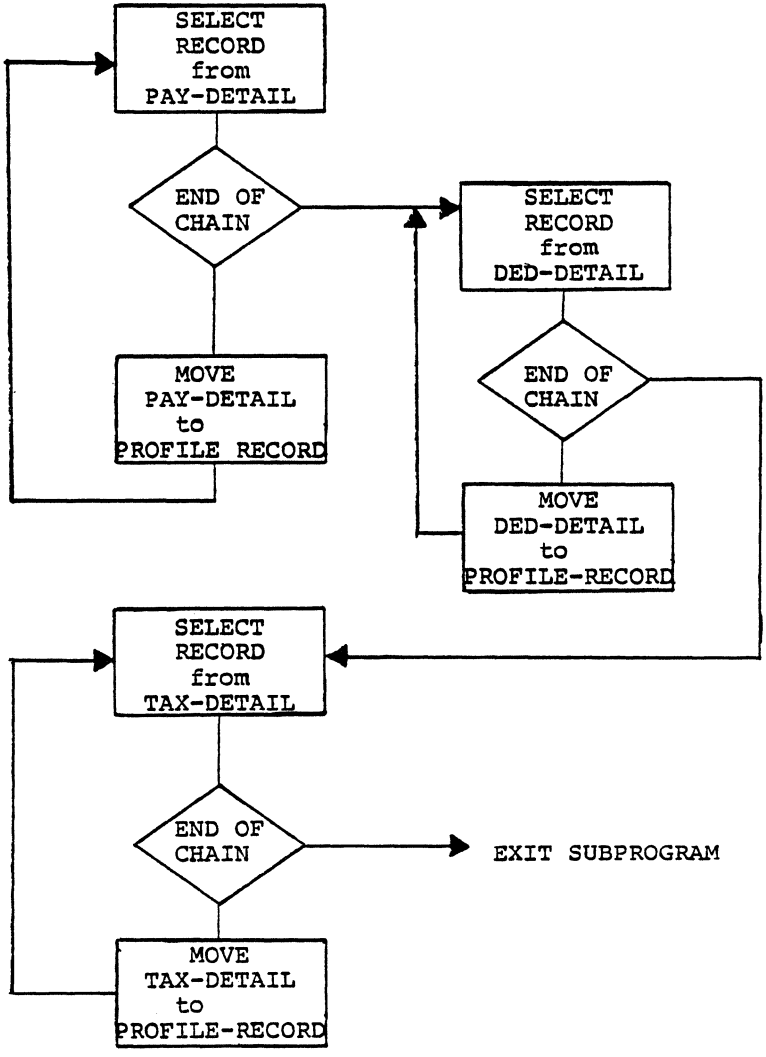
PSEUDOCODE FOR PROFILE PROCEDURE

```
REPEAT
   SELECT NEXT RECORD FROM PAY-DETAIL DATA SET FOR
     SELECTED CHECK-NBR
   IF END-OF-CHAIN
     EXIT
   ELSE
     MOVE PAY-DETAIL DATA TO PROFILE RECORD
   ENDIF
ENDREPEAT

REPEAT
   SELECT NEXT RECORD FROM DEDUCTION-DETAIL DATA SET
     FOR SELECTED CHECK-NBR
   IF END-OF-CHAIN
     EXIT
   ELSE
     MOVE DEDUCTION-DETAIL DATA TO PROFILE RECORD
   ENDIF
ENDREPEAT

REPEAT
   SELECT NEXT RECORD FROM TAX-DETAIL DATA SET FOR
     SELECTED CHECK-NBR
   IF END-OF-CHAIN
     EXIT
   ELSE
     MOVE TAX-DETAIL DATA TO PROFILE RECORD
   ENDIF
ENDREPEAT
```

FLOW CHART - PROFILE PROCEDURE



FREEDOM/SCREEN
A SCREEN PROCESSING SYSTEM FOR ANSI X3.64

JOEL LUEDEMAN
GULF COAST WASTE DISPOSAL AUTHORITY

INTRODUCTION

The Gulf Coast Waste Disposal Authority is a local governmental agency created by the Texas State Legislature.

In 1979, we acquired a Hewlett-Packard 3000 Series III. One of our reasons for choosing the HP was the availability of a screen processing system called V/3000. As we began the implementation of V/3000 based programs, we became aware of the impact the V/3000 intrinsics had on response. During 1981, we developed a replacement for V/3000. This replacement was tailored to an emulation of HP's 264X series of terminals. Our replacement kept one of V/3000's limitations, it would not work on non-HP terminals.

During 1982, we looked for a less expensive alternative to using HP terminals. We adopted the American National Standards Institute (ANSI) X3.64 communications protocol standard. Unfortunately, just as with "standard" COBOL, we found there was no "standard"; i.e., because of the various levels of permitted implementation and vendor extensions, no two "standards" matched. Still, X3.64 did give us a framework to start from.

We evaluated terminals during the first eight months of 1982 and finally selected the TELRAY Model 16 because of the features it had available. Therefore, this version of our screen processor conforms to their implementation of ANSI X3.64.

Though we do not use V/3000 and our routines use different calling arguments, it would be possible to write a V/3000 interface such that V/3000 programs would merely need to be recompiled to use this screen processing system. To that end, we have included a program to allow a V/3000 form to have a migration path to our system, VIEWCOPY.

1.1 OVERVIEW

FREEDOM/SCREEN uses the chain concept of linking forms together. For each form chain there is one form defined as the MASTER form; i.e., the chain head. Each form contains linkage to the master form, the previous form in the chain, and the next form in the chain. A form should only belong to one chain. A chain may consist of only one form, in which case all links point to that form. There may be more than one chain per forms file. See Section 1.5, Advanced Concepts, for a discussion of chain structures.

A forms file may contain up to 85 forms. Each form may contain up to 400 fields. A field consists of one to 80 characters. A form does not have to contain any fields; i.e., a form may contain only instructions or other documentation.

The first record in the forms file is the file directory. Each form in the file requires three records. The first record contains the text portion of the form. The second record defines each field on the form. The third form contains the initial values for each field.

1.2 BUILDING A SIMPLE FORM

The program is executed either by the MPE command "RUN EZBUILD.PUB.SYS" or by a UDC command defined for this purpose.

After entering execution, the program asks a series of questions about the form and the forms file.

1.2.1 Enter the FORMS FILE NAME

Enter the name of the forms file. If no name is entered, execution is terminated and control is returned to MPE. The name of the forms file may consist of up to 35 characters following the MPE file naming conventions. If your file has a lockword, it may be entered at this time, otherwise the system will prompt for it later. If this file exists, it will be "opened" and execution continues, else the folowing message will appear:

File "name as input" is not found!
Should this file be created? (Y or N)

If this file should NOT be created, answer N and execution will return to 1.2.1 allowing you to reenter the file name or exit. Answer Y to create the file.

1.2.2 Do you want to see the FORM DIRECTORY? (Y or N)

If you answer N, execution continues at 1.2.3. If you answer Y, the number of entries in the directory is displayed as well as the name, record location number, batch file record length, and form length in rows for each form in the forms file. The batch file record length will be non-zero only for MASTER forms, in which case it will be the maximum length of a batch record of the chain. This record length will be automatically adjusted upward, as necessary, if a form is added to the chain or a form in the chain is altered. At the end of the directory display, you will be asked:

Do you need to reconstruct the FORMS DIRECTORY? (Y or N)

If you answer N, execution continues at 1.2.3. If you answer Y, you will be asked to:

Enter the FORM NUMBER
Enter the new FORM NAME

Enter the RECORD LOCATION
Enter the RECORD LENGTH
Enter the new MAXIMUM NUMBER OF ROWS

The FORM NAME and RECORD LOCATION may be left blank. If so, they will not be changed. The RECORD LOCATION begins at record two (2) for the first form in the file; the directory is in the first record. The RECORD LOCATION is computed as follows:

RL = 2 + (FN - 1) *3
Where: RL is the RECORD LOCATION
FN is the FORM NUMBER

The RECORD LENGTH and MAXIMUM NUMBER OF ROWS must be entered or they will be set to zero. They may have a value of zero. This procedure is the only way to lower the maximum values for a chain. The sequence of questions will be repeated until a FORM NUMBER of zero is entered. The directory will be redisplayed and you will be asked:

Do you need to reconstruct the FORMS DIRECTORY? (Y or N)

If the directory is empty, a message to that effect will be displayed and execution will continue at 1.2.3.

1.2.3 Enter the FORM NAME

The form name may be up to eight (8) characters long.

If no form name is entered, execution returns to 1.2.1.

If the form cannot be "found," i.e., it is not a preexisting form, you will be asked:

Are you sure you want to ADD this form? (Y or N)

If you answer N, execution returns to 1.2.3. If you answer Y, you will be asked:

Is this FORM to be COPIED from a SCREEN FILE? (Y or N)

If you answer N, execution skips to 1.2.4. If you answer Y, you will be asked:

Is this FORM to be COPIED from an INTERIM VIEW FILE? (Y or N)

If you answer N, execution skips to 1.2.5. If you answer Y, execution continues at 1.2.6.

If this form can be "found," you will be asked:

Do you want to DELETE this form? (Y or N)

If you answer Y, the form will be removed from its chain and execution will return to 1.2.3. The form will NOT be deleted from the forms file, only from its form chain. If you answer N, you will be asked:

Is this FORM to be REPLACED from a SCREEN FILE? (Y or N)

If you answer Y, execution skips to 1.2.7. If you answer N, execution skips to 1.2.8.

1.2.4 Adding a New Form

As each form is added to the forms file, it must be correctly positioned in the forms chain. If the form is a chain containing only itself, all links still must be set. If the form is being added to an existing chain, the links must be set to correctly insert or append the form. If a new chain is being formed, the links must be correctly initialized. EZBUILD attempts to correctly set all linkage for you by asking a series of questions.

Is this FORM one of a CHAINED series of forms? (Y or N)
What is the name of the MASTER FORM for this chain?
Which FORM PRECEDES this form in the chain?

If you answer the first question N, the rest of the questions will not be asked and all links will point to the current form.

After the linkage is created, you will be informed that you may design your form. The function key designations are quite different from those defined by V/3000 (see Figure 1). You will note that at this point you are unable to assign area qualifiers to a field; this will be handled later. Forms are designed as in V/3000, but fields are unnamed. Any value placed into a field will be assumed to be that field's initial value. Line drawing and mosaic character sets may be used to enhance the forms appearance. Inverse video, dim, blink, and other visual attributes of the text portion of the form are supported. There is a limitation of 400 fields per form and 80 characters per field. The only limit to the number of lines per form is the amount of terminal memory. The form is limited to 6000 characters, including escape sequences.

If you wish, for any reason, to start over with the screen cleared and set to default condition, depress function key F5.

When you are finished designing your form, depress the "ENTER" key. This key is on the numeric keypad.

The form will then be analyzed by the program. The form will be separated into three parts; the text portion of the form, the field definitions, and the field initial values.

The program will ask:

Do you want to modify field characteristics? (Y or N)
At which field do you wish to begin modifying?

If you answer the first question N, the second question will not be asked. See Figure 2 for the characteristics which may be changed. You will loop to the second question until you enter a field number of zero (or blank). You may use function key F5 to position to the previous field or function key F6 to position to the next field. You may change any characteristic except the ROW the field is in. Care should be taken when changing the length of the field. Combinations of visual attributes and area qualifiers are allowed. The combinations are not checked for validity. Depress the "ENTER" key when you have made all of the changes for a given field. If you wish to get out of the modification process before handling all fields, depress function key F8.

When all fields are modified as required, the program will ask:

Do you want this screen analyzed? (Y or N)
Do you want a print of this analysis? (Y or N)

If you answer Y to both questions, the analysis will be displayed on your CRT and printed on the system printer. If you answer Y only to the first question, the analysis will be displayed on your CRT but no print will be produced. If you answer N to the first question, processing continues by storing the form on the forms file and returning to 1.2.3.

1.2.5 Copying a Form from a Screen File

Sometimes it is easier to alter an existing form than it is to create a form from scratch. You may copy a form from a different forms file or from the current forms file. The program will display:

Enter the FORMS FILE NAME
Enter the NAME of the FORM to be COPIED

The name of the forms file may consist of up to 35 characters following the MPE file naming conventions. If the file has a lockword, it may be entered at this time, otherwise the system will prompt for it later. The form name may be up to eight (8) characters long.

As with adding a new form from scratch, the positioning of the form in the chain is handled through the program asking:

Is this FORM one of a CHAINED series of form? (Y or N)
What is the NAME of the MASTER FORM for this chain?
Which FORM PRECEDES this form in the chain?

After the linkage is created, you will be informed that you may design your form. The program will "open" the specified forms file and display the form to be copied. You may now make any alterations to the form. If you wish, for any reason, to start over with a fresh, unaltered copy of the form, depress function key F5.

When you are finished designing your form, depress the "ENTER" key. This key is on the numeric keypad.

The form will then be analyzed by the program. The form will be separated into three parts: the text portion of the form; the field definition; and the field initial values.

The program will ask:

Do you want to modify field characteristics? (Y or N)
At which field do you wish to begin modifying?

If you answer the first question N, the second question will not be asked. See Figure 2 for the characteristics which may be changed. You will loop to the second question until you enter a field number zero (or blank). You may use function key F5 to position to the previous field or function key F6 to position to the next field. You may change any characteristic except the ROW the field is in. Care should be taken when changing the length of the field. Combinations of visual attributes and area qualifiers are allowed. The combinations are not checked for validity. Depress the "ENTER" key when you have made all of the changes for a given field. If you wish to get out of the modification process before handling all fields, depress function key F8.

When all fields are modified as required, the program will ask:

Do you want this screen analyzed? (Y or N)
Do you want a print of this analysis? (Y or N)

If you answer Y to both questions, the analysis will be displayed on your CRT and printed on the system printer. If you answer Y only to the first question, the analysis will be displayed on your CRT but no print will be produced. If you answer N to the first question, processing continues by storing the form on the forms file and returning to 1.2.3.

1.2.6 Copying a Form from an Interim V/3000 File

To avoid having to build forms that are already existing as V/3000 forms, we have provided a two-stage process to copy these forms.

The first step requires a HP 264X terminal or emulator. Execute VIEWCOPY, a program which displays the V/3000 form on the 264X terminal and copies it to an interim V/3000 file. Then execute EZBUILD to copy the form from the interim file, display it on the screen, allow it to be altered, then store it in the current forms file. The procedures to be followed are the same as those explained in 1.2.5.

1.2.7 Replacing a Form from a Forms File

On rare occasions you may wish to replace an existing form with one from another forms file. The procedures to be followed are the same as those explained in 1.2.5.

1.2.8. Updating a Form

The most common activity of form manipulation will be form modification. A form may have its position in the chain altered, the text portion of the form may change, the field characteristics of one or more fields may be modified, or any combination of the above may occur.

This process may be followed with no changes made, if all you want to obtain is an analysis of the form.

The program will ask the following questions to allow the position of the form in the chain to be altered:

Do you want to CHANGE this screen's order in the chain? (Y or N)
What is the NAME of the MASTER FORM for this chain?
Which FORM PRECEDES this form in the chain?
Which FORM FOLLOWS this form in the chain?
Are you changing the MASTER FORM for this chain?

By carefully answering these questions, two chains may be joined, a form may be used only for entering or exiting a chain, or a form may be moved within the chain. Please refer to Section 1.5, Advanced Concepts, for a further discussion of form chaining.

After the linkage is handled, the program will ask:

Do you want to MODIFY this screen as displayed? (Y or N)

If you answer Y, the screen will be displayed and may be modified. If you answer N, the form will immediately be analyzed by the program. The form will be separated into three parts: the text portion of the form; the field definition; and the field initial values.

The program will ask:

Do you want to modify field characteristics? (Y or N)
At which field do you wish to begin modifying?

If you answer the first question N, the second question will not be asked. See Figure 2 for the characteristics which may be changed. You will loop to the second question until you enter a field number of zero (or blank). You may use function key F5 to position to the previous field or function key F6 to position to the next field. You may change any characteristic except the ROW the field is in. Care should be taken when changing the length of the field. Combinations of visual attributes and area qualifiers are allowed. The combinations are not checked for validity. Depress the "ENTER" key when you have made all of the changes for a given field. If you wish to get out of the modification process before handling all fields, depress function key F8.

When all fields are modified as required, the program will ask:

Do you want this screen analyzed? (Y or N)
Do you want a print of this analysis? (Y or N)

If you answer Y to both questions, the analysis will be displayed on your CRT and printed on the system printer. If you answer Y only to the first question, the analysis will be displayed on your CRT but no print will be produced. If you answer N to the first question, processing continues by storing the form on the forms file and returning to 1.2.3.

1.3 USING EZENTER

Using EZENTER is similar to using V/3000's entry. The major difference is in using the BROWSE mode. EZENTER allows you to position directly to any record in the batch file. The function keys retain their same meaning as in V/3000, with one exception. In BROWSE mode, F1 does not return the first record of the batch file; instead it allows you to reposition to any record in the batch file. See Figure 3 for the function key definitions.

The batch file created by EZENTER is compatible with those built by V/3000. Application programs using these batch files require NO modification to use a file created by EZENTER.

1.4 USING THE LIBRARY

As part of our screen package, we have provided a library of over 100 routines in the form of an RL. These routines are self-documenting as to purpose and use. All are written in FORTRAN.

Routines beginning with the letters BAS provide an interface for BASIC programs. These routines were necessary because BASIC cannot take advantage of the alternate return path option of a FORTRAN subroutine.

Routines of particular interest are:

Linking	MASTERFORM NEXTFORM PREVFORM REFRESHFORM
Paging	GETPAGE LOADPAGE NEXTPAGE PREVPAGE SETUPPAGEMODE
Positioning	CURSOR HOME POSFIELD

Reading	DEMANDBLOCKREAD GETFIELD GETFIELDASCII GETFIELDDINT GETFIELDDP GETFIELDINT GETFIELDREAL SCNBUFFERLOAD SCREENREAD SCURSOR
Writing	DISPLAYBUFFER EDITSCREEN FRAMESCREEN PUTFIELD PUTFIELDASCII PUTFIELDDINT PUTFIELDDP PUTFIELDINT PUTFIELDREAL
Message	CLEARWINDOW HILITEFIELD PUTWINDOW RESETSTATUSLINE
Editing	CHECKFIELDS CONVERROR REQFIELD UNDEFINEDKEY
Macrokeys/Function Memory	RESETMACROKEYS RESETSOFTKEYS SETMACROKEY SOFTKEY
Printing Screens	PRINTSC PRINTSCREEN PRINTSCRNCLOSE
Dynamic Form Redefinition	GETFIELDDEF REDEFINEFIELD REDEFINEFORM REDEFINESCREEN REDISPLAYFORM

1.5 ADVANCED CONCEPTS

There are four points that need discussion here: field mapping; form linking; paging; and dynamic redefinition of the currently displayed form.

1.5.1 Field Mapping

Fields may be mapped from a master form to a detail form. This is similar to a SAVE field in V/3000. The receiving field should be large enough to hold the data transferred from the master form. A field may be "mapped" to more than one field on the detail form. Mapping is enabled by indicating the master form field number to be mapped to the detail form when altering the detail form's field characteristics (see Figure 2). Mapped data replaces the field's initial value.

1.5.2 Form Linking

A form is linked in three directions: forward; backward; and to the master form. Normally the program checks to insure that all links are in the same chain. However, when you update a form using the procedure in Section 1.2.8, you may have the links point to different chains by answering Y to the question:

Are you changing the MASTER FORM for this chain? (Y or N)

Be careful when doing this. You will have to check each form involved to insure all links are as you intended. This technique, while possible and sometimes necessary, is difficult to implement. A better way to solve this problem is through the multiple pages handling capability available in the terminal.

1.5.3 Paging

The standard TELRAY Model 16 includes four pages of volatile display memory. Optional configurations are eight pages of volatile, or four pages of nonvolatile, or eight pages of nonvolatile display memory. The memory can be logically subdivided by the programmer. A "standard" page consists of 24 lines of 80 characters. The logical line length can be reset to any integer number from 20 through 255 inclusive. The logical page length can be reset to any integer value from 4 to 255 inclusive; however, it cannot be set to a number that would exceed the memory capacity of the terminal. If this is attempted, the terminal will automatically reset the logical page length to the maximum allowed by the memory available. The number of pages can be reset to any integer number from 1 to 255 inclusive; however, it cannot be set to a number that would exceed the memory capacity of the terminal. If this is attempted, the terminal will automatically reset the number of pages to the maximum allowed by the memory available.

The library contains several routines that will allow the programmer to configure the terminal memory (SETUPPAGEMODE), load forms into specific pages of terminal memory (LOADPAGE), position specific pages of memory on the display (GETPAGE), and position among the pages of memory (NEXTPAGE and PREVPAGE).

Each page of memory may contain a different form chain. NEXTFORM, PREVFORM, REFRESHFORM, and MASTERFORM operate on the page of memory currently displayed. By using the paging facility provided in the terminal, complex chaining schemes should not be necessary.

1.5.4 Dynamic Redefinition of the Currently Displayed Form

In an effort to minimize the apparent delay in terminal response, FREEDOM/SCREEN "paints" each form in one pass. The text portion of the form is combined with the field definitions and written to the screen when the form is first displayed or when the form is refreshed. If a form "repeats," the text portion is not rewritten to the screen; instead, all fields are cleared to their initial values.

For a given application, multiple forms may differ only by column headings or row descriptors. These differences are generally thought of as being part of the form text. If these areas could be made protected fields and if there were a method of dynamically changing their initial values, multiple forms would not be necessary. This concept is called FRAMING.

The programmer, by using routines provided in the library, has the ability to dynamically control the nontext portion of the form.

FREEDOM/SCREEN uses three buffers to assign initial values to a field. SBUFFER contains the initial value defined when the form was created. MBUFFER contains data to be "mapped" from the master record to the field. FBUFFER contains data provided by the programmer to replace the initial value of the field. The data in FBUFFER, if nonblank, replaces the data in SBUFFER. When a field is "mapped," the data in MBUFFER replaces the data in SBUFFER, no matter what data is in FBUFFER. SBUFFER is written to the screen when the form is initially displayed or when it is refreshed. FBUFFER is used to change the subsequent field values. FRAMESCREEN is called to refresh the nontext portion of the form.

Sometimes it would be helpful if a field's characteristics, visual attributes and/or area qualifiers, could be dynamically altered during program execution. For example, have a date field change from unprotected to protected once the date was entered. Again, the library provides routines to allow this. GETFIELDDEF returns the current form's current field characteristics. REDEFINEFIELD and REDEFINESCREEN redefine the field characteristics currently being displayed. REDEFINEFORM redefines the current form's field characteristics. REDISPLAYFORM provides the same function as REFRESHFORM, but without rereading the form's file, thus allowing the redefined field characteristics to remain in effect.

1.6 Configuring the Teleray Model 16

The manufacturer has provided two means of resetting to the initial state:

```
escape c
escape [255;255 space y
```

Neither do exactly what we need. The escape c merely clears internal buffers. Escape [255;255 space y resets to the manufacturer's default conditions, NOT the conditions specified by the user.

We have evolved a procedure to configure the terminal for use on the HP-3000. First, reset to default conditions, escape (255;255 space y, then execute CONFIGUR.PUB.SYS.

This configuration process totally redesignates the 64 macro keys provided by the manufacturer (see Figure 5). A "soft" reset is provided by depressing key 33; i.e., line local. This "soft" reset essentially returns the terminal to unprotected, character mode operation and is an adequate reset after "aborting" from block mode.

FUNCTION KEY DEFINITIONS DURING FORMS DESIGN

f1	Beginning of Field
f2	End of Field
f3	Line Drawing Character Set
f4	Normal Character Set
f5	Panic Escape When an Error Is Made During Forms Design
f6	Undefined
f7	End Area Visual Attributes
f8	Undefined
f9	Undefined
f10	Dim On
f11	Undefined
f12	Underscore On
f13	Blink On
f14	Undefined
f15	Inverse On
f16	Blank (i.e., No Echo)
f17	Undefined
f18	Undefined

Figure 1

FIELD CHARACTERISTICS DEFINITION FORM

Field ---

Field Length in characters	---	
Type of field	-	1 - Transmit only 2 - Unprotected
Field position	---	Row
	---	Column
Mapping	---	

Visual Attributes		Area Qualifiers	
Dim (half bright)	-	Numeric only	-
Underscore	-	Alphabetic only	-
Blink	-	Right justified	-
Inverse video	-	Left justified	-
Blank (no echo)	-	Zero filled	-
		Must fill	-
		Entry required	-

Figure 2

Key	Data Collection Mode	Key	Browse/Modify Mode
MASTER FORM f1	Display the master form for this chain	BROWSE MODE f1	Allows the user to position to any record in the batch file
f2		DELETE REC f2	Delete current batch record from the batch file. Note that you cannot insert a record in place of a deleted record; any new records are added to the end of the batch file.
PRINT f3	Print current form on line printer. Prints form with current data (but not any values typed on screen and not yet entered by ENTER).	PRINT f3	Same as in Data Collection
REFRESH f4	Clear screen; initialize terminal; and redisplay with initial values. (Can be used to restore form should it accidentally be cleared from the screen by a local RESET or terminal power failure. If terminal power fails, a colon prompt is issued when power returns and you must type RESUME before pressing REFRESH.)	REFRESH f4	Same as in Data Collection except that previously entered data is displayed.
f5		PREVREC f5	Display previous record in batch file on form used to enter data.
NEXT FORM f6	Display next form	NEXT REC f6	Display next record in batch file on form used to enter data.
BROWSE f7	Enter browse/modify mode	COLLECT f7	Return to data collection mode.
EXIT f8	Exit from EZENTER; return to MPE control.	EXIT f8	Exit from EZENTER; return to MPE control.

Figure 3

FIELD DEFINITION OF EACH FIELD

1 - Field Length in Characters
2 - Type of Field
    1 - Transmit Only
    2 - Unprotected
3 - Row Field Is In
4 - Column Where Field Begins
5 - Mapping Indicator
    If this form is a detail (i.e., not a master) form and there is a relationship between this field and a field on the master form, this word contains the field number of the field on the master form

6 - Field Visual Attribute

Bit	Definition	Decimal	Octal
0	Unused	32768	1
1	Unused	16384	4
2	Unused	8192	2
3	Unused	4096	1
4	Unused	2048	4
5	Unused	1024	2
6	Unused	512	1
7	Unused	256	4
8	Blank	128	2
9	Inverse Video	64	1
10	Unused	32	4
11	Blink	16	2
12	Underscore	8	1
13	Unused	4	4
14	Dim (half bright)	2	2
15	Unused	1	1

7 - Field Area Qualifiers

Bit	Definition	Decimal	Octal
0	Unused	32768	1
1	Unused	16384	4
2	Unused	8192	2
3	Unused	4096	1
4	Unused	2048	4
5	Unused	1024	2
6	Entry Required	512	1
7	Must Fill	256	4
8	Transmit Only	128	2
9	Unused	64	1
10	Zero Fill	32	4
11	Right Justify	16	2
12	Alphabetic Only	8	1
13	Numeric Only	4	4
14	Unused	2	2
15	Left Justify	1	1

8 - Character Number Where the Field Begins

Figure 4

For the macro key locations see the figure on page 3-4 of the TELERAY MODEL 16 Instruction Manual

Key	Value	Key	Value	Key	Value	Key	Value
01	?p	19	7	33	reset	51	one upper left
02	?q	20	8	34	insert character	52	one up
03	?r	21	9	35	delete character	53	one upper right
04	?s	22	roll up	36	insert line	54	unused
05	?t	23	4	37	delete line	55	one left
06	?u	24	5	38	erase page	56	home
07	?v	25	6	39	erase line	57	one right
08	?w	26	roll down	40	unused	58	unused
09	?x	27	1	41	unused	59	one lower left
10	?P	28	2	42	unused	60	one down
11	?Q	29	3	43	unused	61	one lower right
12	?R	30	0	44	unused	62	unused
13	?S	31	.	45	unused	63	calculator mode
14	?T	32	CR	46	unused	64	block transmit
15	?U			47	+		
16	?V			48	-		
17	?W			49	*		
18	?X			50	/		

Figure 5

Note that the function keys (01-18) are never used to enter data; data is entered only with the ENTER key.

IMAGE/3000: Designing for Performance & Maintainability

Wendy M. Matheson

Hewlett Packard (Canada) Limited
6877 Goreway Drive, Mississauga,
Ontario, CANADA L4V-1M8
(416) 678-9430

ABSTRACT

This paper addresses design alternatives for implementation of efficient data base applications using IMAGE/3000 as the Database Management System (DBMS). Included will be discussion on design considerations, performance, and maintainability.

INTRODUCTION

Your database design is finished. Whew. You breath a sigh of relief, relax, and think of all the work that has been done. You have taken the IMAGE/3000 training, evaluated the application requirements, spoken to the end users, spoken to management. You have arranged the data to satisfy the needs of all these people. Result: A schema. This morning you corrected the last syntax error in the schema, ran it successfully through the schema processor (DBSCHEMA), and created the database for the first time. All you have to do now is give the go ahead to begin the application development.

STOP! Wait. What have you really done?
Designed a database of course.
Will the users get what they asked for?
Yes, of course.
Will the users get what they expect?

It turns out that users have high expectations from database applications. It's no wonder, they are continually being told how wonderful databases are. Databases certainly ARE wonderful, but NOT foolproof, not yet anyway. Your database design can result in your company's enjoying the benefits of database, or not.

So what do the users EXPECT?

End users will usually expect 'user friendly' software and 'all the data at their fingertips', whatever that means. If you probe further you will probably get someone to admit that they expect the Data Processing Department to become more receptive to changing requirements because it is going to be so easy to accommodate. (Take this person aside and make sure they tell you what they're planning to change!) Your management have been reading TIME, Business Week, etc. and will probably think that all data will magically be transformed into INFORMATION, whatever they define information to be. Your programming staff will be looking forward to programs that are easy to write and maintain, and, to 'not thinking about how the data is stored'. Everyone will expect good performance now, and in the foreseeable future.

High expectations indeed.

Are you getting worried? I would be.

The most difficult step is the database design. The key to success is a good database design, the first time. *"Determining how the database should be structured so that it meets the present information requirements and also is flexible enough to withstand the inevitable changes caused by the business environment is more of an art than a science."*(Reference 1) An ART? Yes, that about covers it. There just aren't 20 fixed rules you can follow to guarantee success, only guidelines, experience, and talent.

Get out that copy of the schema and write on it in big letters; PRELIMINARY.

Finalizing a schema and giving the go-ahead to start application development is kind of like getting married. The minute you do it, it becomes very difficult and very expensive to change your mind. You want to make sure that your design will satisfy your end users needs and expectations. You want to make sure you haven't made any design decisions that, unbeknownest to you, will cause problems somewhere down the line.

Okay, what now? How about reviewing the data base design. Invite the design team (if there's more than just you) and a few other interested parties who haven't seen the design lately. (You've been looking at that design for so long, you are probably the worst candidate for finding problems.) Make sure one of the people reviewing the design has a sound technical knowledge of IMAGE/3000 and is up to date on enhancements, etc. If not, invite a consultant from Hewlett Packard. The cost of consulting upfront is probably insignificant compared to the cost of fixing a problem that you didn't catch later on in the development cycle.

Review the design, consider performance (everyone seems to ignore performance until they don't get any), then send the schema and related procedures on to program development.

The remainder of this paper is a list of guidelines that should be considered during a database design review. For each topic, my objective is to point out what factors should be considered, pros, cons, and alternatives. The outcome should be a database design which is both functional and will provide a solid basis from which to write efficient and maintainable applications.

GUIDELINES FOR REVIEWING AN IMAGE/3000 DATABASE DESIGN:

SECURITY

A good place to start is the database security, passwords, access types, etc. (If you haven't done the security better postpone the design review and do it.) It is an excellent way to review the functionality of the database and spot problems.

If User Type One needs access to eight different data sets to get at the eight data items needed to do the function, the data locality in the database needs work. You want a single functions' data in as few sets as possible.

If you have given an online function WRITE access to a detail dataset which is linked to eleven masters, you have just created a potential performance problem. (I'll discuss paths in more detail later.)

Reviewing the security reminds us of what the individual functions will be doing.

Is there a password for every user type or function expected on the database? (Not too many passwords, wouldn't want that to prevent you from changing them regularly...) Setting up separate passwords for batch processing is also a good idea.

For each password give the minimum access necessary (for security reasons) at the highest level possible (for performance reasons). Use item level security only when necessary.

LIMITS

If the schema passes through DBSCHEMA without errors then you haven't REACHED any limits, but, are you approaching any? Check for potential problems where you are NEAR to a limit. Following are some of the current upper limits in IMAGE/3000 that you should watch out for: (as of version 03.05)

```
Data Item Names per Data Base : 255
Data Item Names per Data Set  : 127
Data Sets per Data Base       :  99
Detail Data Sets per Master   :  16
Search Items (keys) per Detail:  16
Maximum entry size            : 4094 bytes
Maximum # of entries per Data Set: 8,388,607
Maximum # of entries per chain    :    65,535
```

Make sure that now, and in the future, it does not appear that these limits will cause a problem with your data base design. (If you would like to see any of these limits increased, just fill out a Service Request for the enhancement and send it in. The more the merrier!)

PATHS

For every path in the data base you should have a reason. A good reason.

The cost of a path is in the maintenance of the pointers (and in disc storage, but that is minor). This cost is incurred on adds (DBPUT) and deletes (DBDELETE).

Ask yourself the following questions for each path:

1. *How much will it be used?*

All the time? *Great!*
Most of the time? *Good!*
Overnight batch runs? *Are you sure it's worth it?*
Monthly? *You must be joking...*

2. *Will it mean faster access?*

I hope so! Ideally you are putting the path in for this reason.

3. *How often are adds and deletes done?*

If the data never changes you only incur the overhead of the path when loading the data. If adds and deletes are done, what is the volume? Consider putting in a data item to be used as a delete flag. The program could 'logically' delete the record

by setting the flag ON, then the actual deletes could be done either in a background processing or at night.

4. *Will the key value ever change?*

If YES, then watch out. You will be doing a DBDELETE and a DBPUT to satisfy this requirement.

5. *Will the longest chain be less than 10% of the total number of entries in the (detail) data set?*

If your chains are going to be a significant portion of the total entries in the set, then perhaps you should read the set serially instead of maintaining a chain.

The general rule is the simpler the better. Avoid the temptation of putting in alot of automatic masters because it *'might be nice'* to have that path. For good performance, you want just a many paths as you really need, no more.

SORTED CHAINS

Yes, there are good reasons to have sorted chains, and there are good reasons not to.

The difference between a sorted and an unsorted chain is the method used to do the DBPUT.

On DBPUT to an unsorted chain IMAGE will always add the entry at the end of the chain. For sorted chains, IMAGE starts at the end of the chain and works backwards through the chain until it finds the correct location based on the value of the sort item, then IMAGE inserts the entry. If IMAGE finds a duplicate sort item value, it then extends the sort field to the end of the entry and effectively does a multifield sort. (If you want IMAGE to do this unsolicited 'multifield sort', great, else it's good idea to make the sort item the last item in the data set to supress this.)

For these reasons, the cost of a DBPUT to an unsorted chain is fairly constant, but, for a sorted chain it depends on where the entry will be put.

The worst case would be a long sorted chain with lots of additions in random places in the chain. On the other hand, if the sort value always increases (IE. date:time) then there is almost no additional overhead involved in having the chain sorted.

Some good reasons to use sorted chains:

1. *When you must maintain an order.*

NEVER depend on being able to do a DBUNLOAD,CHAINED. Many people have fallen into the trap of saying; *'IMAGE adds my data in chronological order so I don't need to worry about maintaining the order myself.'* The day will come (says Murphy) that you will be looking at a Serial DBUNLOAD and saying *'but I can't afford to lose the order...'*. If you must maintain an order and don't want to use a sorted chain, at least keep the data needed to re-create the order in the data set. Then, if Murphy comes to visit, you can write a program to re-create the order.

2. *When you seldom (or never) add.*

3. *When the sort values are always ascending.*

4. *When the chain is short.*

5. *When an interactive process requires it.*

6. *NOT for reporting!*

If you have decided to use sorted paths pre-sort the data before loading the database, and for batch processing.

PRIMARY PATH

Always choose a primary path for each detail data set.

By default, IMAGE will select the first unsorted path, as shown in the schema. If all paths are sorted, IMAGE selects the first sorted path. Even if the defaults are satisfactory, indicate the primary path (with an !) in the schema so others know you have selected that particular path.

Then what happens?

Nothing.
Unless you do DBUNLOAD/DBLOAD's periodically. When you do a DBUNLOAD,CHAINED (default), the detail entries are unloaded in primary chain sequence. The DBLOAD then puts the entries back in this order. Result: The primary chain entries are together on disc... in the same blocks... read in more than one at a time... great stuff for performance when reading down the primary chain! If the chains are long you get a significant performance improvement!! Go for it!!!

Which path should I select?

The most benefit will come from selecting a path that is accessed frequently, by interactive users, and has whose average chain length is fairly long. There is little benefit in selecting a path whose chains are only one or two entries long.

KEY SELECTION

IMAGE/3000 uses primary address calculations to place entries in master data sets. If two different key values have the same primary location they are called 'synonyms'. When a synonym is encountered, IMAGE finds the closest available location in the dataset and places the record there, linking it to the record in the primary location. This is called a 'synonym chain'. The more key values that have the same primary address, the longer the synonym chains, and the more overhead for chain maintenance (on the DBPUT).

When a program calls DBFIND for a key value, IMAGE does the primary address calculation and reads in the block containing the primary location. If the requested entry isn't in the primary location IMAGE uses the synonym chain to find the entry. Long synonym chains can result in several blocks being read in to find a single entry instead of just one.

All this happens behind your back.

For best performance, master data sets should have as few synonym chains as possible. The factors affecting this are the data type of the key, the key values, and the capacities selected for the master data sets.

The data type you select for each key determines what type of algorithm IMAGE/3000 will use to calculate the primary addresses.

For master data sets with keys which have data types U, X, Z, or P, IMAGE/3000 uses a hashing algorithm which approximates a uniform distribution of primary addresses in the master data sets, regardless of the bias of the search item values. To achieve the minimum number of synonyms the capacity of the data set should be a prime number, and should be large enough that the set will be, at most, 80% full.

For master data sets with keys which have data types I, J, K, or R, IMAGE/3000 uses modulo arithmetic, based on the data set capacity, to calculate the primary address for the key. (This algorithm is described in the IMAGE/3000 Reference Manual, Reference 2.) The number of synonyms that will be generated using this algorithm depends heavily on the search items values and the data set capacity.

EXAMPLE ONE: If you have a numeric key, Order Number, whose values range from 1000 on up (1000, 1001, 1002, ...), and a capacity of 1001, there will be no synonyms!

EXAMPLE TWO: If you have a numeric key, Part Number, whose values range from 1000 on up (1000,1001, 1002, ...) for Product Line A, and from 2000 on up (2000, 2001, 2002, ...) for Product Line B, with a capacity of 1001, the second group of key values will have the same primary locations as the first! Terrible for performance!

If you are using numeric keys, check the algorithm described in the manual so that you can avoid situations as described in Example Two. Often, by selecting the right capacity, you can prevent groups of key values from generating the same primary addresses, although you may end up with a much larger capacity than you planned. Other alternatives include changing the data type and changing the data. These may (or may not) be realistic alternatives depending on the customer.

CAPACITIES

When selecting capacities for data sets remember that:

1. IMAGE allocates ALL disc space when you create the data base.

2. Changing the capacity normally requires a DBUNLOAD/DBLOAD which can't be done at a moments notice.

For master data sets, follow the guidelines presented in the previous section.

In detail data sets IMAGE will take the capacity specified and change it slightly to make most efficient use of disc space, based on the blocking factor. Space occupied by deleted entries is re-used. For detail data sets there are no performance considerations for capacity, just pick a capacity large enough so that you don't force an unscheduled DBUNLOAD/DBLOAD.

BLOCK SIZE

IMAGE/3000 uses a default maximum blocksize of 512 words to select the blocking factors for each data set in the data base. In general the blocksize selected by IMAGE is a GOOD CHOICE!

If you decide to optimize the blocking, set out to improve the main activity on the database. If most activity is random, consider smaller blocks. If most activity is sequential, consider larger blocks. Remember that IMAGE uses the largest block size in the data base to determine what the buffer size will have to be. Having one data set with a very large block size will force large buffers, and waste main memory.

The maximum block size is overridden using the $CONTROL BLOCKMAX= command. To set a different maximum for each data set, just put a $CONTROL statement before each set definition in the schema.

TRANSACTIONS

Here it is.
The real test of your dedication.
To get a feeling for the performance of the data base application, you must define the different logical transactions that will be used, and exercise them.

What is going to happen when the order entry clerk hits 'ENTER'?

The application program will execute a series of CALL statements to IMAGE Intrinsics to process the order.

How many CALL statements?
To which intrinsics?

Walk through each type of transaction and see how it works with the data base design. Watch out for transactions that do a large number of DBPUT's and DBDELETE's. (To be most effective, locking and recovery strategies should be completed, and the necessary intrinsic calls included in the transactions.) If there are any areas of concern, consider testing the performance of the transactions against the data base design.

There are some utilities available to help with the transaction testing:

DBDRIVER: This utility is part of the IMAGE/3000 product, and, unfortunately, undocumented. Documentation is available from the Users Group (Reference 11), or, try calling your friendly Account Systems Engineer! The DBDRIVER utility allows you to set up intrinsic calls to any IMAGE/3000 data base, executes the calls, and gives you the elapsed time (along with other interesting information).

IMAGE Database Evaluative Analyzer (IDEA): IDEA is available through the Users Group Contributed Library. This utility allows you to define transactions, number of processes, and think time, and develope a script that simulates the data base activity of your application. The results include estimates of response time and transaction throughput. (Reference 13)

These types of tools assist in comparing different types of transactions, estimating minimum response times, and comparing the performance of various data base designs. Once these types of tests are set up, they can be run several times. A good way of estimating the impact of a design change (example: adding an extra three automatic masters) is to run a simulation with and without the change.

DATA BASE CREATION

Most people will say that the creation of an IMAGE/3000 data base is a simple two step process:

1. *Run DBSCHEMA to create the root file.*

2. *Run DBUTIL,CREATE to create the data sets.*

That's all there is to it. No problem. Except, for performance reasons, add two more simple steps to the procedure, and do them every time you create a data base.

3. *Run DBUTIL and set the buffer specifications according to the applications requirements.*

The buffers for an IMAGE/3000 data base are kept in the Data base Control Block (DBCB) and shared among the users accessing the data base. The default buffer specifications that IMAGE uses have changed over time so it is very important that you set them yourself, to ensure you get the correct number.

To set the number of buffers, use the >SET BUFFSPECS= command in DBUTIL:

EXAMPLE: >SET BUFFSPECS=32(1/120)

This will instruct IMAGE to create the DBCB with 32 buffers for anything from 1 to 120 users accessing the data base.

EXAMPLE: >SET BUFFSPECS=8(1/2),9(3/4),10(5/6),11(7/8),12(9/120)

This will instruct IMAGE to start out with 8 buffers, and for every two additional users add a buffer. Terrible for performance! Every time the number of buffers has to be changed, the DBCB size has to change, causing a SWAP.

GENERAL RULE:

```
BUFFSPECS=20(1/120)  for small memory, or small number of users
BUFFSPECS=32(1/120)  for average application
BUFFSPECS=64(1/120)  for DBLOAD type programs
                     (yes, it will run faster !!)
```

For very large applications, running on large memory systems use the formula in the IMAGE Reference Manual (Reference 2) to calculate the approximate size of the DBCB, and then estimate the maximum number of buffers and set based on that. (Remember that the DBCB can be, at most, 32,767 Words, and leave some room for the trailer area to expand.)

4. *Decide which disc each data set will be placed on, and then place them accordingly.*

IMAGE/3000 will build the data base using the default device specification of 'DISC'. This will result in the data sets being spread accross all those devices which have device class DISC.

For best performance, separate masters and related details, and spread the heavily accessed data sets over the available devices. To do this, STORE the data base, and then do selective RESTORE's using the DEV= parameter. Once this has been done, future RELOAD,SPREAD's and RESTORE's will put the data sets back where you requested.

CAUTION: If you are planning to do IMAGE transaction logging (*great idea!*) to disc, DON'T put the transaction log file on the same disc(s) as the data base!!! If you have a disc hardware problem, you are going to want to recover at least one or the other.

DATA BASE MAINTENANCE

Once the data base is loaded and the applications are up and running, priorities shift to maintenance.

Maintaining the integrity of a data base:

Is the data still correct?
Is the structure still correct?

The design stage should include putting in place procedures to ensure that if the data base developes an integrity problem, it is detected as soon as possible. (For an excellent discussion on this subject see Alfredo Rego's article; *DATA BASE THERAPY: A Practitioner's Experiences.*) Procedures should also be developed to detect potential performance problems (example: long synonym chains in Master data sets).

Some of the available utilities to help in these areas include:

DICTDBA: This utility is included as part of the DICTIONARY/3000 product. DICTDBA (Data Base Audit) reports on the usage statistics and checks internal linkages of a data base. Statistics include information on synonyms (master data sets) and chains (detail data sets). (Reference 12)

DBLOADNG: This utility is available through the Users Group Contributed Library. DBLOADNG produces a report including information on synonyms and chains. (Reference 5)

DBCHECK: This utility is available through the Users Group Contributed Library. DBCHECK checks the internal structure of a data base.

The other major part of data base maintenance is keeping up with changing requirements. Things do change, requirements

change, priorities change, business practices change, and data base designs should change along with them. Structural changes to data bases are a natural part of the maintenance of a data base application.

There are a number of different ways to make structural changes. Which you choose will depend on what it is you want to change, time constraints, and, of course, money constraints. Some techniques for making structural changes include:

DBUNLOAD/DBLOAD: These utilities are included with the IMAGE/3000 product. The allowable structural changes are clearly detailed in the IMAGE/3000 Reference Manual. (Reference 2) Read them carefully!

CUSTOM PROGRAM: You can always decide to write a program to unload your data base (or a portion of it) and load it back up just the way you want it!

DICTDBU/DICTDBL: These utilities are included with the DICTIONARY/3000 product. The data base load utility (DICTDBL) compares the old schema with the new schema and, if necessary, transforms the data field formats of the old data base to the format of the new data base. (Reference 12)

ADAGER/3000: ADAGER/3000 (The Adapter/Manager for IMAGE/3000 Data Bases) is a proprietary software system available from Rego Software Pty (Reference 14) which provides facilities to make structural changes. (Reference 10)

AND SO ON

The subject of data base design is an ongoing one. IMAGE/3000 and related products change over time. Opinions change over time (mine certainly have!).

Remember that everything has a cost and a benefit. It is important to consider all factors and get the most up to date information available when designing a data base (or when doing anything else, for that matter). Several good articles have been written on this subject and several opinions presented. I am looking forward to many lively discussions on this material at the Users Group Meeting!

Thank You.

REFERENCES

1. Finneran, Thomas R. & Henry, J. Shirley, *Structured Analysis for Database Design*, DATAMATION, November 1977, Volume 23, No. 11

2. IMAGE Data Base Management System Reference Manual, HP3000 Computer Systems, Hewlett-Packard Company, Part No. 32215-90003, Sept. 1979

3. Blasgen, Michael W., *Database Systems*, SCIENCE, Feb. 12, 1982, Volume 215 No. 4534, American Association of the Advancement of Science

4. Ross, Ron, *Database Technology Forgets Managers*, Computer World, Volume XIV, No. 52, Dec. 22, 1980, Computer World Communications Inc.

5. Bergquist, Rick, *Optimizing IMAGE: An Introduction*, Journal of the HP3000 International Users Group Inc. Second Quarter 1980, Vol. 3, No. 2

6. May, Jim, *Programming for Performance*, Journal of the HP3000 International Users Group Inc. July/December 1982, Volume 5, No. 3,4

7. Fennelly, Rich, *Database: A Slippery Concept, Hard to Define*, Computer World, April 9, 1979, Computer World Communications Inc.

8. Rego, F. Alfredo, *DATABASE THERAPY: A Practitioner's Experiences*

9. Greer, David J., *IMAGE/COBOL: Practical Guidelines*, Robelle Consulting Ltd., Proceedings; NAUG San Antonio 1982

10. Rego, F. Alfredo, *IMAGE'S COMING OF AGE: Breaking free from restrictions to Data-Base transformations.*

11. *DBDRIVER: Did You Know You Had It?*, SMUG II Micro-proceedings, Robelle Consulting Limited, San Antonio, March 1, 1982

12. DICTIONARY/3000 Reference Manual, HP3000 Computer Systems, Hewlett-Packard Company, Part No. 32244-90001, July 1982

13. IDEA/3000 IMAGE Data Base Evaluative Analyzer Version 3 User's Manual, June 1978, HP3000 Users Contributed Library

14. ADAGER/3000, Rego Software Pty, Calle del Arco 24, Antigua, GUATEMALA

PROGRAMMING FOR PERFORMANCE

This paper represents my personal concepts of programming and performance as best I can express them at this time. Nothing I have either included or excluded is meant to represent the official posture of the Hewlett-Packard Company.

Hewlett-Packard allows, expects, and encourages individual initiative. The existence of this paper exemplifies that philosophy in action. Hewlett-Packard has completely supported my efforts. I have enjoyed total freedom in my choices of subject, content and format.

This paper originated from the Baltimore, Maryland sales and service district office of the Eastern Sales Region. It was physically prepared on equipment in the Technical Center in Rockville, Maryland.

The graphics were created on a 2647A terminal using Interactive Formating System software. Textual data, sample programs excepted, were prepared using HPSLATE. Final printing was done on the 2680 Laser Printer. A COBOL program using IFS intrinsics generated input to the printer.

I have had much help and cooperation in preparing this paper. Thanks to all with special appreciation to Ruth, my wife, for her patience, support and unselfish sacrifice of many evenings and weekends during its preparation.

Jim May

SECTION 1 GENERAL INTRODUCTION

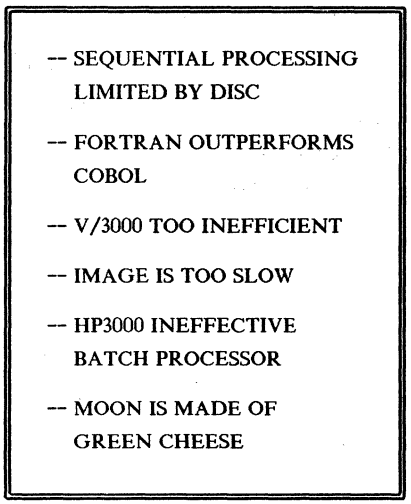


Figure 1

TRADITION (Figure 1)

-- SEQUENTIAL PROCESSING IS LIMITED BY DISC ACCESS

This is one of those generally accepted truths that all of us have been taught from the beginning of our careers. In most cases, on most machines, this is probably a valid generality. Because HP has concentrated on transaction processing, certain defaults built into the file system often make this assumption invalid when evaluating batch processing performance.

-- FORTRAN OUTPERFORMS COBOL.

In typical business applications, this is generally untrue. The FORTRAN compiler is admittedly superior in handling numeric data if we limit the definition of numeric data to binary and real data formats. The COBOL compiler, however, does a much better job with ASCII numeric data and is quite effective when dealing with files, records, and ASCII character fields. The net effect is that COBOL usually outperforms FORTRAN in the average commercial application.

-- V/3000 IS TOO INEFFICIENT.

Considering the powerful, highly generalized capabilities provided within V/3000, this statement is not acceptable. The original V/3000 did have design characteristics, particularly the use of KSAM and the prohibitive form file recompilation techniques, that left a bad taste in our mouths. The new V/3000 has corrected these shortcomings. In any particular application, a good programmer could probably outperform V/3000; even so, V/3000 is a highly efficient subsystem that deserves the chance to earn our confidence.

-- IMAGE IS TOO SLOW.

IMAGE is not at all slow but it is easily abused. IMAGE is an excellent example of a network data base and as such is inherently very rapid when used for record retrieval. Unfortunately, the price of rapid random retrieval is relatively slow structure maintenance, particularly when the structure becomes complex. Add to this the overhead for sophisticated internal security, multiuser update access, and extensive chain sequencing and you have a heavily burdened environment. In effect, the demands of our applications lead to slow performance; IMAGE itself is not inherently slow.

-- HP 3000 IS INEFFECTIVE IN BATCH PROCESSING.

The HP 3000 running MPE is admittedly biased towards transaction processing. The defaults built into MPE have not been chosen to maximize batch processing; the programmer who attacks a batch application without taking this into consideration may be displeased with the results. An informed programmer will know which override options to invoke in order to bring out the batch processing strengths of the machine. When handled properly, the HP 3000 is capable of surprising batch performance.

-- THE MOON IS MADE OF GREEN CHEESE.

I finally stopped believing this in 1969.

PROGRAMMING	FOR	PERFORMANCE
-- Programmer is problem solver		-- Measured by your MANAGEMENT !!
-- Programming is problem solving		-- Compromise between elements
-- Entire problem is fair game		-- Doing job -on schedule -within budget
-- Entire solution is fair game		-- Response (TP & B)
		-- Throughput (TP & B)

Figure 2

PROGRAMMING AND PERFORMANCE (Figure 2)

-- PROGRAMMING

A programmer is much more than a technican who encodes a problem solution in machine readable form. A true programmer is deeply involved in formulating the solution to the problem and, in some cases, may even identify and define the problem prior to compounding a solution.

No discussion of programming, therefore, can be limited strictly to an examination of computer coding techniques. The subject must be broadened to include all aspects of the problem and all details pertinent to the problem solution.

-- PERFORMANCE.

We technicians often forget that the criteria for performance measurements are defined by management and may differ greatly between organizations and even between functional areas within an organization. In too many cases we refuse to accept the fact that the only satisfactory solution may require compromise, most often a sacrifice of technical elegance, in order to meet a management objective. A technically advanced solution finished too late may be worthless; one that exceeds the planned cost may be even worse.

Fortunately for us technicians, this discourse will concentrate on technical performance. We will be concerned with traditional indicators, response and throughput, in both batch and transaction processing environments. Many times, one can be gained only at the expense of the other. Luckily, some techniques can improve performance in all instances.

PERFORMANCE CURVE (Figure 3)

A generally acceptable graphic depiction of a machine's performance is a curved line, the "performance curve", showing the gradual performance degradation for the average program as the machine is progressively loaded. In typical transaction processing environments, the curve shows a definite pattern (Curve #1). At first, performance degrades very little as the first few interactive jobs compete with one another. As more jobs are added, each tends to have a

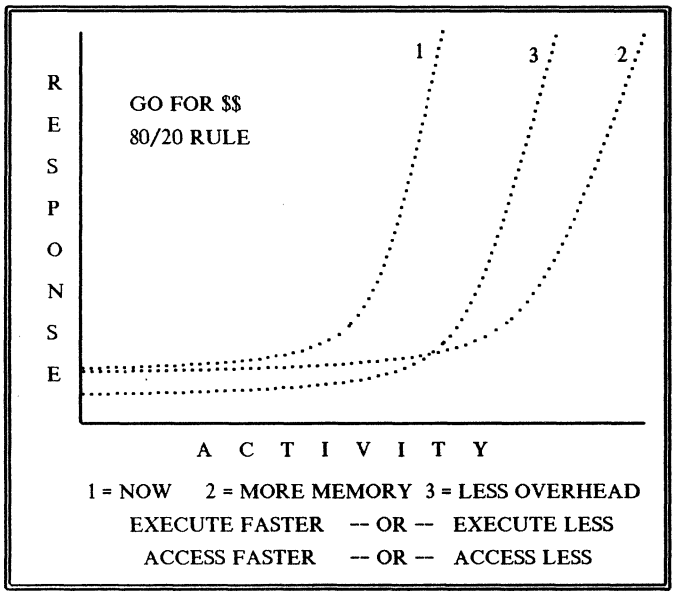


Figure 3

greater negative impact and the curve begins to climb more rapidly. Eventually the "knee in the curve" is reached where each added job causes a disproportionate degradation. At the knee the machine has usually reached the point where the aggregate useful work done by the machine drops for each job introduced.

If the real memory in the machine increases, performance normally improves as shown by Curve #2. On the low load end, the performance improves only slightly if at all since low load performance is not generally memory limited. Mid-range performance improvement is more noticeable and the mid-range itself is extended. The knee still occurs but does not show up until the machine is more heavily loaded. The usual impact of additional memory shows up more in improved throughput rather than in improved individual program

response.

Increasing the raw execution power of the machine affects performance differently from increasing its memory. Improvement shows up immediately on the low end of the curve and the knee shows up later. Similar improvement can also be attained by improving mass storage access even without increased raw execution power. Curve #3 shows a typical example of such improvement.

Improved programming exhibits characteristics similar to those of Curve #3. We could expect this since improved programming usually causes less code to be executed or less disc accesses to be made. This, in fact, becomes a basic guideline for improving programming. Most effective performance improving techniques center around reducing executed code and reducing disc access. Since improvements in disc access are more practical to accomplish and will reduce code execution as a byproduct, disc access reduction usually assumes first priority.

TRANSACTION PROCESSING PERFORMANCE (Figure 4)

Batch performance is fairly easily measured. We can easily time how long a job runs, what resources it seems to be absorbing, and how much competing batch jobs inhibit one another. TP performance is much more difficult to quantify.

TP performance is measured by two yardsticks, throughput and response. Throughput is the objective count of the number of transactions that can be processed in any given time period. Response is more difficult to measure because it is a subjective evaluation.

A programmer has more influence on response than on throughput. Additionally, dramatic changes to response may make little change to throughput. This paper will concentrate on response.

TP is too complex to try to examine as a single entity. For simplicity, I am limiting this overview to an evaluation of

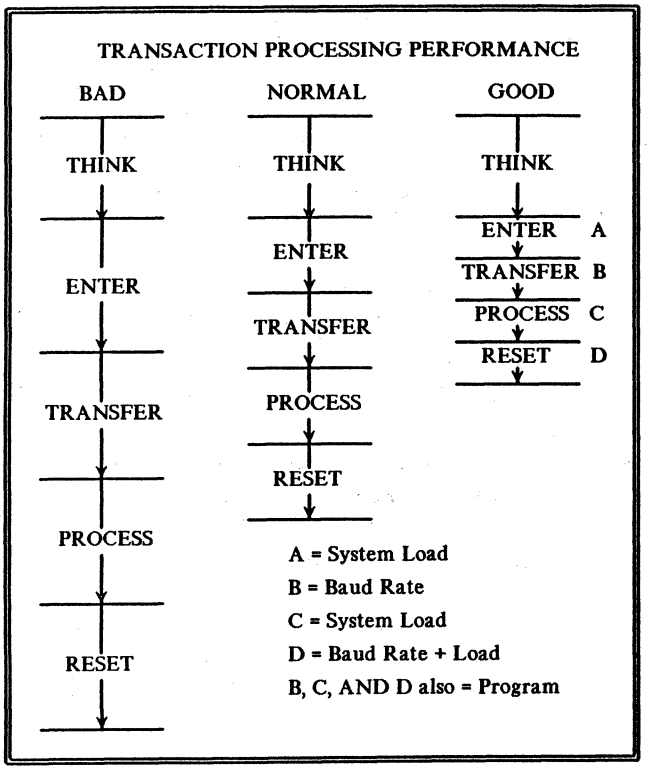


Figure 4

the processing of a simple transaction. This can be broken into 5 distinct pieces of time.

1. Think Time.

The time for the user to fill a screen. Think time is normally the longest item in TP and is application dependent. I will arbitrarily disregard think time in this paper.

2. Enter Time.

The time from the hitting of the enter key to the beginning of actual data transfer to the machine. This time is mostly determined by the hardware, the operating system, and the overall machine load. Although a programmer may impact enter time, we will not discuss it.

3. Transfer Time.

The time from the start of data transfer from the screen to its completion. This is mostly a hardware function. In some instances the programmer can change this item. We will address transfer time briefly.

4. Process Time

The time your program spends actually processing the screen input. By definition this is under programmer control. We will concentrate on improving process time.

5. Reset Time

The time required to prepare the terminal for the next user interaction. In pure data entry this time might be trivial. In other applications it might involve screen switches, response transmission or similar lengthy activities. We will examine some cases involving reset time.

Traditional measurements of system response times tend to emphasize the measurement of enter time. For system evaluation this is a valid point of reference. We programmers must concern ourselves with application dependent factors over which we have some control.

Programmers deal most directly with transfer, process, and reset times. These three items are the primary criteria used by users in measuring how responsive a system is.

The user who waits fifteen seconds for the computer to digest a screen of data and prepare the terminal for next input feels that the machine is not responsive. Our job is to make

the machine responsive in the eyes of the user.

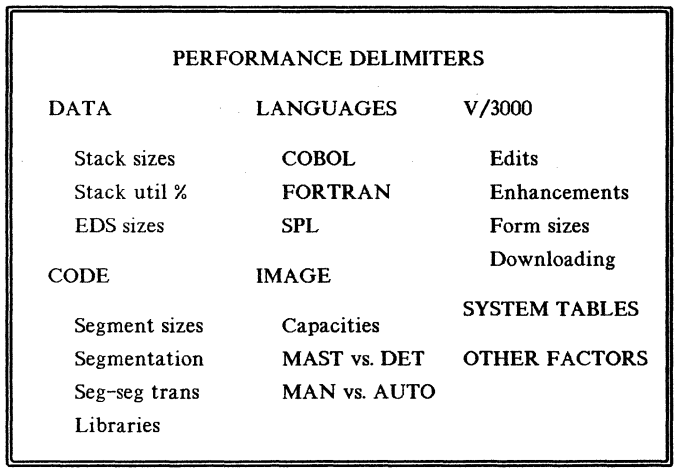


Figure 5

PERFORMANCE DELIMETERS (Figure 5)

Almost anything can impact performance. I have consciously oversimplified the situation by dividing the subject into seven groupings. Each of these groupings could be considered most important by any individual. This would be influenced considerably by a person's experience and background. A quick overview of each will set the tone for the presentation of my views on the subject.

GROUPS 1 AND 2 -- DATA AND CODE.

These related items are very important factors influencing performance. They have also received much attention by many people.

This is fortunate in that the programmer has definite

guidelines to follow to try to avoid creating totally unacceptable programs. It is unfortunate in that we often assume that the volume of verbiage on a subject indicates its relative impact on our work. I feel this is not always true and that some of us have become entirely too concerned with data and code considerations.

I am almost certain that neither data nor code are the primary culprits when the first questions I am asked about a program are "Do I have my stacks small enough?" or "Have I segmented the program properly?". All too often these are dead giveaways that the programmer has become intimidated by the massive documentation about code and data and has failed to get a good perspective on the whole situation. How, for example, could code and data be causing 10 second response delays in a program that makes 250 data base accesses per response?

I firmly believe that code and data can significantly impact performance. I believe even more firmly that they should be held suspect only after many of the other possible contributors have been reviewed and evaluated.

GROUP 3 -- LANGUAGES.

This is another area where I feel we have all expended too much effort. Except for special cases in which specific capabilities of a language are required, I have yet to find a program which would be meaningfully more efficient in one language than another. The one rule I would accept would be the prohibition of interpretive BASIC in a production environment.

I have seen many instances where the absolute requirement to have a "COBOL shop" has denied a programmer access to efficiencies available in other languages, particularly SPL. I have seen even more cases where the fear of the assumed inefficiencies of COBOL has resulted in FORTRAN or SPL programs which quite often run slower than a COBOL version.

Although this may represent a minority viewpoint, I claim

that COBOL will often be the most efficient and the most effective language in the typical commercial application. And even if this may not be true, I further claim that the choice of languages does not have a meaningful impact either way. If you like COBOL, use it. If you hate COBOL, use FORTRAN or perhaps SPL. In any case, use whatever helps you as an individual to get your job done.

GROUP 4 -- IMAGE

By now, everyone should be wondering just what I think are the important performance delimiters. We've finally hit one. IMAGE has a big effect on performance.

The factors sublisted in this group are the ones usually quoted as being critical. These and the use or non-use of sorted chains have been discussed to death. In addition, with the probable exception of the constraints on sorted chains, their importance is generally blown all out of proportion.

We will cover the impact of IMAGE but it will be done from a different vantage point. Our primary concern will be centered around the ways we have structured our data bases and the effects those structures have on performance.

GROUP 5 -- V/3000.

V/3000 is very good but it is not perfect. As with any highly generalized package, V/3000 can probably be "beaten" in any given application by a highly skilled programmer. In some applications, V/3000 may also be a less desirable choice because more performance oriented but less generalized techniques are available. A case in point might be the data capture environment where we would want to consider using the more efficient data capture intrinsics.

Just because V/3000 may not always be the best solution in all instances should not become a rationale for avoiding V/3000 altogether. V/3000 simply has too many capabilities to be ignored. It is also more efficient than most of us

probably realize.

V/3000 was introduced with two specific faults which gave some of us some bad memories. Both of these have been corrected. The KSAM oriented forms file structure has been replaced by a vastly superior file access method. The ability to recompile only modified forms has greatly reduced development and maintenance overhead. If we have not reviewed our evaluation of V/3000 since its introduction we may be cheating ourselves.

There are some capabilities still suspect within V/3000. I claim that in most cases, the culprit is the heavy demands we build into our applications rather than the way V/3000 handles those demands. I strongly suspect that most of us would be pleasantly surprised at the performance V/3000 gives compared to that provided by user written code performing the same functions. We would also probably be appalled by the volume of code we would have to write and maintain to replace standard V/3000 capabilities.

There is much potential benefit for us if we closely evaluate our techniques of using V/3000. In many cases we can improve performance by using V/3000 differently. In the average application, however, we would probably do much harm by trying to avoid or replace V/3000.

GROUP 6 -- SYSTEM TABLES.

Let's handle this fast. Look at system tables from an overall system point of view. Forget about them in applications programming.

GROUP 7 -- OTHER FACTORS

When somebody ends a list of items with a group called "other factors", they probably plan to quickly dismiss those same factors as relatively unimportant. In our case, I have deliberately placed them last for emphasis. What many might consider relatively unimportant are the very items experience has taught me to look at more carefully. I think that look

will be most revealing.

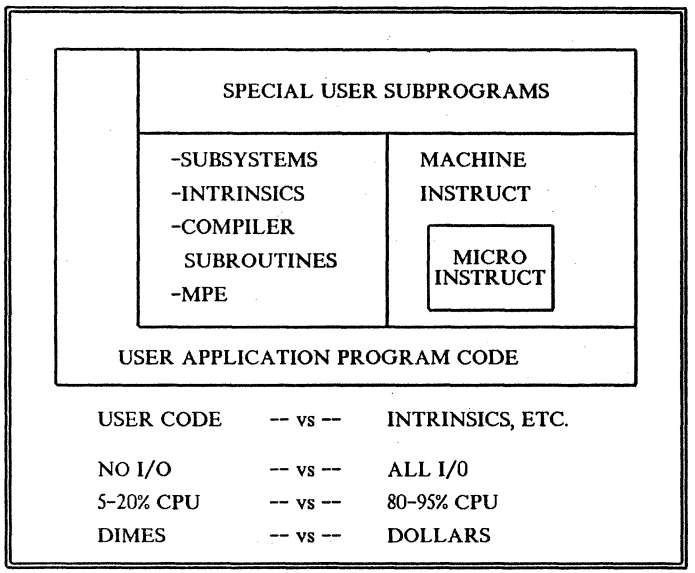


Figure 6

PROGRAM CHARACTERISTICS (Figure 6)

How we visualize our programs can have a great influence on our attempts to improve their performance. A reasonably accurate picture of a typical commercial application program might surprise some of us. It might also help explain why two programs, one written by a highly technical programmer and the other written by an experienced but relatively non-technical programmer, can have maddeningly (to the technical programmer) similar performance characteristics.

The most important fact to be realized is that the normal application program written on a typical modern commercial

computer is a "driver". Whenever we write a program, particularly in a high-level language, we do not generate computer instructions. We are actually writing compiler instructions which will be converted into computer instructions. Our choices of coding techniques can have significant influence on some of the code generated. In general, however, the problem being coded has a far greater effect than our choices of how we code our solution.

A consequence of the "driver" aspect of our programs is that our program code (that is, the portion controlled by our coding techniques) never performs I-O. Except for rare, highly specialized, privileged mode applications, all I-O is performed by MPE Intrinsics. Once we have designed our application characteristics, the I-O required is essentially independent of our programming language or our programming techniques.

Some programmers could be somewhat discouraged at being told their coding techniques have relatively little effect on a program. Others will be relieved on hearing the same message because it allows them to code without fearing that they might mess up a program through "poor" programming. Both of these types of programmers have missed the boat on performance.

Better programmers reprioritize their efforts away from mere technical coding competence and concentrate on design. They realize that the characteristics designed into an application are the major determinants of performance.

Being better programmers, we will concentrate on design in our search for improved performance. Even relatively small changes in design can have more effect than massive changes in pure coding. Wise decisions during application design can have immense impact on eventual performance.

SUBPROGRAMS (Figure 7)

Utilizing subprograms allows us to program for performance. Period. End of sentence.

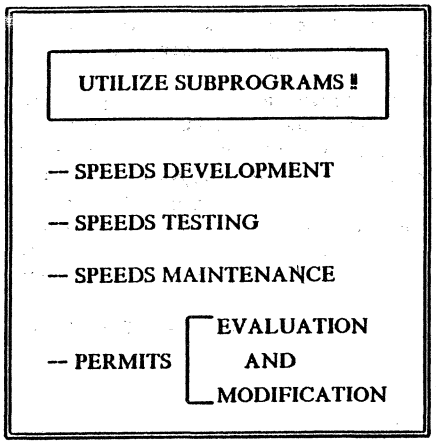


Figure 7

A non-trivial program written using subprograms will be faster to develop and test than an equivalent program written as a single unit. This is not prejudiced conjecture by me; it is a widely recognized fact. Since development and test time for a program are included in the broad definition of performance, programs written in subprogram form give improved performance.

Maintenance is also simplified for programs built from subprograms. It is easier to determine where to modify a subprogram than where to modify a unit program. The validity of the change is more easily tested in a subprogram. Subprograms even require less expense to test than do unit programs. Smaller listings, shorter compilations, and more controllable logic give us tremendous return for our investment.

Subprograms would justify themselves solely on the merits claimed up to this point, but we should look deeper. Subprograms can be powerful tools in the attempt to improve application performance.

Subprograms are prime examples of modular program and application architecture. Modularity isolates functions so that modifications affecting those functions can likewise be isolated. Isolation of modifications allows more accurate evaluation of the effects of those modifications. The more accurately we evaluate our modifications, the more effective our modifications can become.

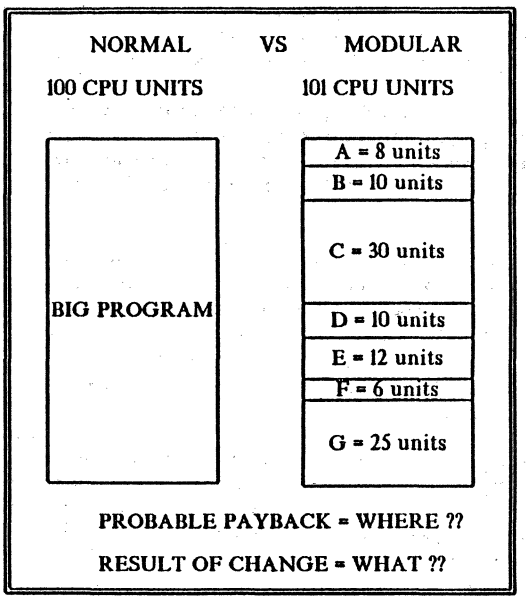


Figure 8

MORMAL vs MODULAR (Figure 8)

Subprograms do not give us more efficient programs. All factors being equal, a unit program will execute more efficiently than one written in subprogram form. Subprograms, however, give us the means to improve performance and efficiency in ways generally unavailable in

unit programs.

Improving performance comes only partly from improving programming. Far more important than how much we improve programming is where we decide to try to improve programming. I would much prefer to reduce a high overhead item than a low one.

Intelligent attempts to improve performance require a disciplined plan of action. That plan must include at least four discreet items:

-- Evaluation of existing performance
-- Identification of candidates for alteration
-- Selection and implementation of changes
-- Evaluation of resultant performance

To get a feel for the potential importance of subprograms for performance improvement, we can review two hypothetical cases. The first involves a unit program; the second, a modular program. In both cases, someone with clout has decided that the performance needs improvement.

CASE I -- THE UNIT PROGRAM.

The programmer follows a rational plan:

-- Measures performance of unmodified program. 100 CPU units are needed for control run.
-- Uses past experience to identify probable bottlenecks
-- Program changes, all quite valid, to improve performance at selected areas
-- Measures performance of modified program. 75 CPU units are needed for test run.

Now we have a few questions to answer:

Q. Was 25 CPU unit improvement good, fair, or poor?
A. Can't tell.

Q. How much of the potential improvement was realized?

A. Can't tell.

Q. Did every change improve performance? A. We don't know. We couldn't make evaluations of each change because the compile cost was too high.

Q. What do we try next? A. Whatever the boss says.

This would have been so easy had the original changes taken us from 100 CPU units to 20. That type of improvement gets praised, not questioned. But who said everything was easy?

CASE II -- THE MODULAR PROGRAM.

This programmer also follows a rational plan:

-- Measures performance of unmodified program. 101 CPU units are needed for control run. Usage per module ranges from 6 to 30 units.
-- Decides that most probably payback is in modules C and G with 30 and 25 unit loads, respectively
-- Uses past experience to decide that module G is most likely candidate.
-- Program changes to module G
-- Measures performance of modified program. 86 CPU units are needed for test run. Module G has gone from 25 units to 10 units.

This programmer also must answer some questions:

Q. Was a 15 CPU unit improvement good, fair, or poor?
A. Quite good. The portion changed showed a 60% (from 25 units to 10) reduction.

Q. How much of the potential improvement was realized?
A. For module G, probably most of it. But we only changed 25% of the program. 75% still merits evaluation.

Q. Did every change improve performance? A. Looks probable.

Q. What do we try next? A. Module C is a likely candidate because it absorbs 30 units during execution. That's more than a third of the remaining overhead!

We may require a number of iterations before we reach the point of diminished returns. At least we have a better way to tell where we have probably reached it. In addition, we have begun to build a body of experience to help us optimize our next program more easily.

SECTION 2 BIG ITEM CHECKLISTS

Talking about techniques to use in isolating performance problem areas is valuable. Knowing what to do next is equally important. This paper will attempt to identify some of the most frequent "next steps".

Sometimes, of course, the "next step" becomes the only step. This occurs most often when we are called upon to help optimize a unit program whose performance is suspect. We can't waste time wishing the program could be more easily analyzed. After all, if it were easily analyzed, we wouldn't have been called in. So we take it as we find it.

Every experienced performance consultant has a mental checklist of potential problem areas. This checklist includes specific techniques found helpful in the past and the expected benefits for each.

Each checklist is different. The differences depend upon the consultants background, track record, and personal biases. A specialist experienced in commercial applications has a different checklist from a specialist who has worked with technical applications. Similarily, checklists based on batch applications will differ from those written for on-line systems.

My background is in commercial applications, both batch and on-line. I would like to share part of my checklist with you. The sequence is for convenience and continuity; it has no priority implications. For each major item, I will organize its analysis this way:

-- A header visual showing:

- TASK: The description of the checklist item.

- PLAN: The proposed corrective action most likely to succeed.

- GOAL: The expected benefits.

-- One or more subordinate visuals showing.

- Why the item might be degrading performance.

- Detail examination of the proposed action.

- Why the proposed action should improve performance.

TASK: REVIEW SEQUENTIAL FILE PROCESSING

PLAN: REBLOCK FILES
USE NOBUF I-O

GOAL: REDUCE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 9

SEQUENTIAL FILE PROCESSING (Figure 9)

Almost every shop has batch programs that process large sequential files. Quite often they are programs originally

written for Brand-X and converted to run on the 3000. They frequently run much slower than we think they should.

We keep reminding ourselves that most commercial applications are I-O bound, not CPU intensive. We begin wondering about the power of the 3000 when our "I-O Bound" programs run at close to 100% CPU utilization. It's time we found out why this happens.

TASK: Check out sequential files looking for high record volumes, low blocking factors, and default file access.

PLAN: Increase blocking factors and replace default file access with nobuf I-O.

GOAL: Reduce disc activity (blocking factors). Reduce CPU load (nobuf I-O).

I-O -- BUFFERED vs. NOBUF (Figure 10)

Those of us who learned programming on a Brand-X machine, know how sequential files are processed. A pair of buffers are set up in the program and the physical I-O system tries to keep them full. The logical I-O system provides records to us by indexing through the buffers as I-O is requested.

This indexing through internal buffers is extremely efficient. Machines using this approach to sequential processing usually handle batch processing better than on-line processing.

Every operating system on every computer is optimized for a particular environment. This includes both the internal architecture of the I-O system and the choice of standard defaults for its user interface. HP emphasizes on-line processing and has designed its I-O system accordingly. Batch processing is performed well but suffers somewhat to benefit on-line work.

On-line processing emphasizes random retrieval. Random retrieval implies retrieval of a record from a reasonably

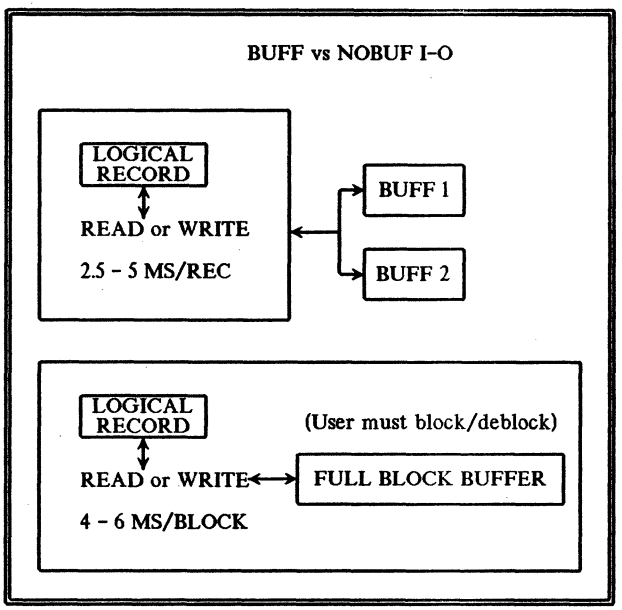


Figure 10

well-defined location on disc. Knowing a records location eliminates most of the benefits of large blocking factors. Therefore, HP has set up relatively small default blocking factors. These utilize disc space well but are usually comparatively small for batch processing.

On-line processing also requires effective file sharing capabilities. Files are quite difficult to share when the I-O system puts the buffer inside programs. HP simplifies file sharing by isolating the buffer from the program. This is excellent for on-line activities but increases overhead when doing batch.

File buffers reside in extra data segments under MPE. Logical I-O requires the file system to expend considerable

effort to transfer records back and forth between the user stack and these extra data segments. This explains the high CPU load during sequential file access.

Once we know how the defaults in MPE increase overhead in batch applications we can make intelligent adjustments. The rewards are well worth the effort. In typical cases, we can reduce overhead by 70 to 90 percent.

Blocking factors are easily changed by coding the "REC" parameter in the "BUILD" command. I can't tell anyone what factors to use but I would probably choose between 7 and 30 depending on record size.

Conversion to nobuf I-O is not so simple but contributes most to CPU load reductions. There are two basic ways to do this:

-- Code your routine directly into your program. Although this is how I coded my sample program, I prefer the second technique.

-- Code your routine in a subprogram. I prefer this technique. It suits my mode of operation.

RECORD SORTING (Figure 11)

Every shop needs sort capabilities. Batch applications are particularly heavy users of sorts because they are inherently sequence dependent.

Sorting places heavy demands on the machine. Although we cannot reasonably eliminate sorting from our programs, we have techniques to reduce their overhead.

TASK: Review our use of the sort capability in our environment.

PLAN: Avoid file-to-file sorts and the use of the stand-alone sort subsystem.

TASK: REVIEW USE OF
SORT SUBSYSTEM

PLAN: AVOID SORT SUBSYSTEM
AVOID FILE-FILE SORTS

GOAL: REDUCE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 11

GOAL: Reduce I-O activity in sort functions. As a secondary benefit, reduce CPU load on the system.

BRUTE FORCE (Figure 12)

When you had a small machine and tape was your primary storage media you learned how to drag data through programs using brute force. The standard mode for sorting was to read a tape into the sort and write sorted records back to tape. Sometimes this had to be done in multiple passes with multiple tapes. It wasn't much but it certainly beat loading and unloading card hoppers.

Modern computer systems support and use tape but they rely more often on disc for primary storage. Modern programmers have also begun to rely on disc. They have finally gotten rid of their little drawers full of punched cards. Why, then, do they still do their work, particularly their sorts, by brute force.

The stand-alone sort is useful and necessary. It is also a resource hog. The I-O required for a sort is extensive, especially when added to the I-O to read and write output

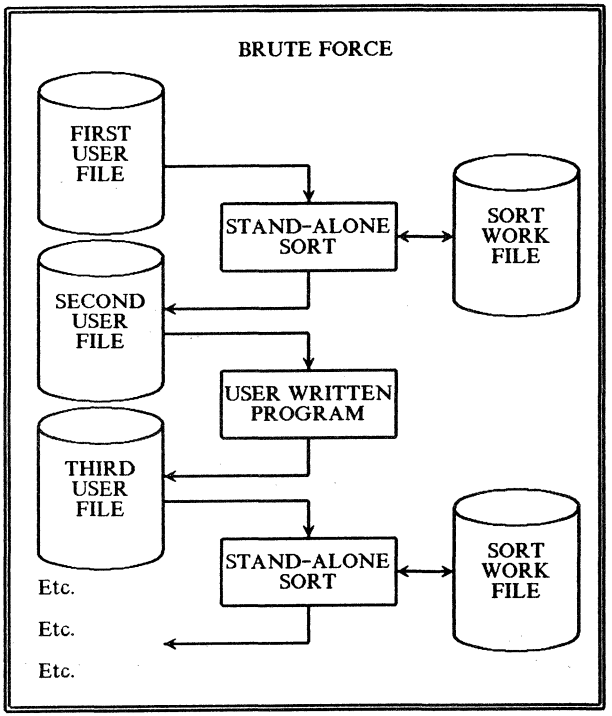


Figure 12

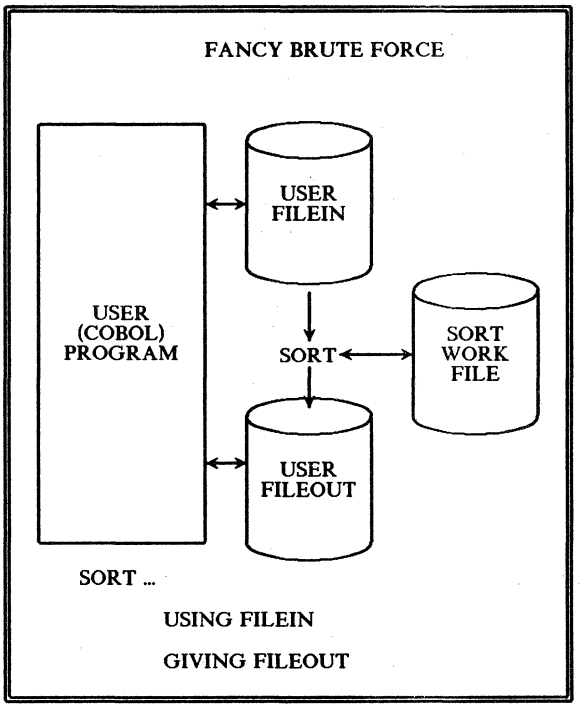


Figure 13

files.

Disc availability is a prime constraint on performance. Sorts, particularly the stand-alone file-to-file sort, eat deeply into this availability. In the interest of performance we should try to reduce these activities whenever practical.

FANCY BRUTE FORCE (Figure 13)

Programmers have learned how to invoke the sort programmatically. They have effectively used this to replace two or more programs and one or more sorts with single programs. Then they have held back from using the true capabilities of programmatic sort access.

Programmatic file-to-file sorts are not necessarily bad but they usually have a negative effect on performance. They use up disc resources at at least the same rate as stand-alone sorts. Except for very small data volumes, programs written with file-to-file internal sorts usually run slower than the

individual programs and sorts they replaced.

Some shops will not allow programmers to use sort capabilities programmatically. I think they are missing a good opportunity by this blanket condemnation. On the other hand, if they are only avoiding "fancy brute force", they can be partially excused.

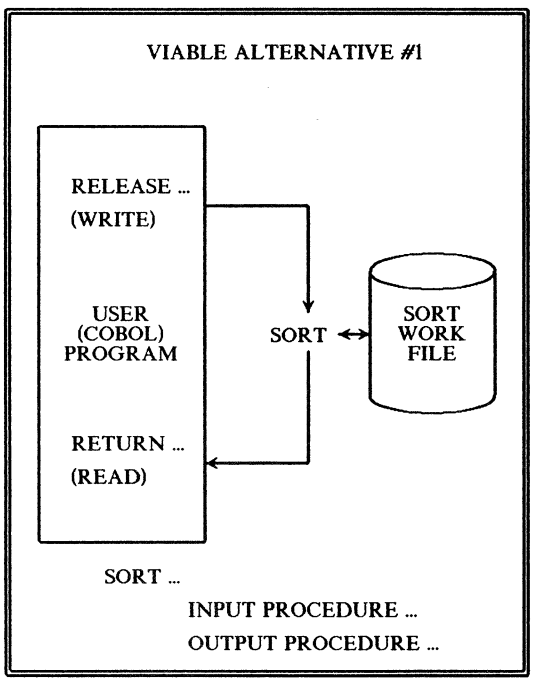


Figure 14

VIABLE ALTERNATIVE #1, DIRECT SORT INTERFACE (Figure 14)

Programmatic access to the sort subsystem gives the

programmer many attractive options. Hooks built into the sort allow programmers to pass records directly to the sort and receive sorted records directly from the sort. This capability can be used effectively to improve program performance.

Direct interaction with the sort allows us to avoid disc activity. Every time we interact directly with sort we avoid two potential disc accesses; we have eliminated a read access and a write access. This can greatly reduce disc activity.

We also reduce CPU overhead by talking directly to the sort. We get rid of the CPU overhead for the file system to process our logical I-O. Unfortunately, there is a price to be paid for this ability.

Interaction with the sort requires direct resources other than disc I-O and CPU cycles. The sort needs memory to be efficient and that memory comes from the user stack. If our stack is too small or the program data is large the internal sort may lose its value. It may then become a burden.

Another limitation of the internal sort is the inability to have multiple sorts executing simultaneously. In programs with multiple internal sorts we may have to allow some file-to-file sorts or the logical equivalent.

VIABLE ALTERNATIVE # 2, PROCESS HANDLING (Figure 15)

MPE offers us another means to reduce sort overhead. We can use process handling as a means to bypass I-O in sorting applications.

Some shops fear process handling. I wish more of them could begin using it to advantage. Perhaps there is too much emphasis on the "special" in "special capabilities". Whatever the reason, many of us look upon process handling as a tool only for exceptional cases. We should view it as an exceptional tool useful in many applications.

There is no justification for reserving process handling

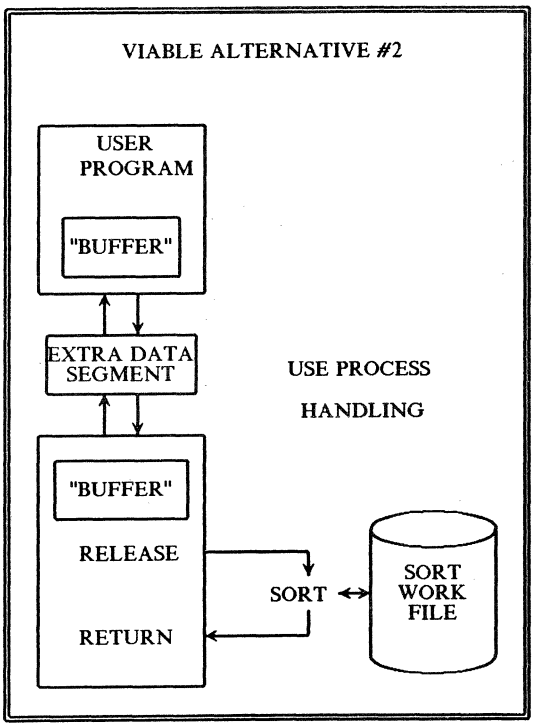


Figure 15

solely for esoteric or multi-threading environments. Process handling is perfectly suitable for use in relatively simple, single-threaded applications. An obvious use would be in a program involving sorts where we can use multiple processes to advantage.

We should examine how sort functions can be accomplished using process handling. The concept is similar to the normal programmatic sort interface especially if we isolate all calls to the process handling intrinsics in a subprogram. The rest is simplicity itself.

1. The master program initiates sorting by a call to the subprogram. The subprogram creates and activates a slave program whose job is to sort records. The subprogram also creates an extra data segment to use in passing blocks of records to the slave. It waits until the slave program is ready for work.

2. The slave begins life by starting up its own internal sort. Its subprogram then wakens the master, in effect saying "OK, I'm ready". It waits for instructions.

3. The master sends raw records to the slave for sorting. Every call to the master subprogram represents logical passage of a record to the slave. The subprogram fills an internal buffer with records.

4. When the subprogram has a full buffer it loads it into the extra data segment and wakes the slave, effectively saying "OK, give these to sort". It waits until the slave has done its work.

5. The slave retrieves the extra data segment, unloads the logical records, and passes them to the sort. When finished, it wakes the master, saying "OK, I'm ready for more". It waits for more.

6. Steps 3, 4 and 5 are looped through until all records have been sent to the slave. The master then sends an "end of data" message to the slave after preparing to receive sorted records. The master waits now.

7. The slave receives the "end of data" message and lets actual sorting begin. When sorted records are available, the slave is ready to awaken its master.

8. The return of sorted records from master to slave is a reversal of the process described for sending unsorted records from master to slave.

A sample program is included with this paper to show the technique. It does not use subprogram interfaces because I wanted to isolate the example in one source file. In real

life, I would recommend a subprogram.

This example obviously reduces disc activity. It offers other benefits as well.

1. The slave has its own stack which is not loaded with application data. The sort can be given plenty of room to breathe. The slave can perform actual sorting more efficiently than could its master.

2. The program is no longer limited to having only one sort active at any time. Except for the normal constraints of MPE, any number of sort slaves may be active at any time.

As with other alternative techniques, process handling extracts a cost.

1. Process handling absorbs CPU overhead. If records are not passed back and forth in blocks, the overhead may be relatively high.

2. Multiple processes and their stacks need memory. This may cause problems in some cases. Your machine size and workload profile are critical decision criteria.

3. Process handling is not difficult but is more complex than the use of standard compiler features. Subprograms eliminate this problem once you have the subprograms written and tested.

DATA VALIDATION AND CONVERSION (Figure 16)

Both batch and on-line programs perform extensive data validations and conversions. Quite often these account for a high portion of the overhead within a program.

Validations and conversions come in two basic flavors- algorithmic and tabulated.

-- Algorithmic validations check validity according to a processing rule. Check-digit calculations and pattern

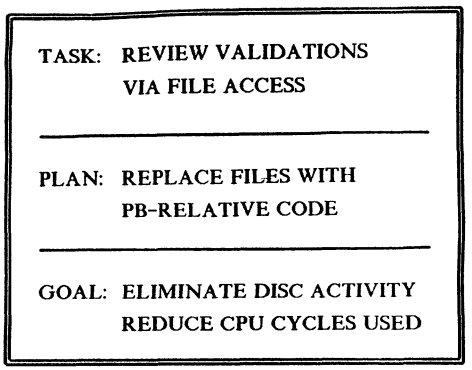


Figure 16

matches are examples.

-- Algorithmic conversions convert data based upon a conversion algorithm. Julian to Gregorian data conversion is an example.

-- Tabulated validations check validity by searching a table or file for a "hit". Customer validation through attempted retrieval against a data base is an example.

-- Tabulated conversions convert data from argument to result by searching a table or a file. Converting numeric error codes to meaningful error messages is an example.

Algorithmic techniques and table oriented searches are normally efficient but may be difficult to modify and maintain. File oriented techniques are easily modified but absorb considerable overhead. Performance considerations may make file oriented techniques too expensive.

TASK: Review file oriented validations and conversions.

PLAN: Replace files with PB-relative code. This normally takes the form of a binary search procedure.

GOAL: Eliminate disc I-O completely when possible. Significantly reduce CPU overhead.

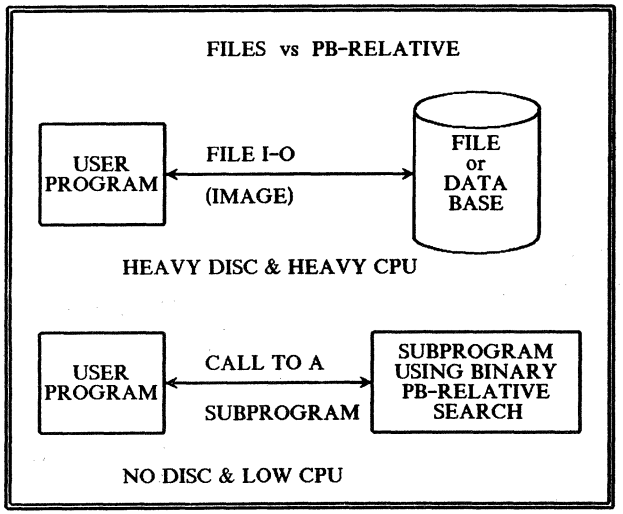


Figure 17

FILES vs PB-RELATIVE (Figure 17)

Files are frequently used for data validation or code expansion in all data processing. We usually use KSAM or IMAGE. In either case the disc I-O is extensive and involves considerable overhead.

In many cases there is no viable alternative to file access. Volatile information is ideally suited to randomly accessed and maintained file structures. Massive quantities of data cannot be economically maintained except on disc media.

In other cases we may be able to use techniques that require little or no disc access. The following are the normal techniques used.

1. Tables may be hard-coded into storage if the information is not too massive and is not volatile. Massive data volumes are prohibitive and volative data values create maintenance hang-ups.

2. Volatile tables may be filled at execution time via file access. This is impractical for large files and may cause excessive overhead in short duration programs.

3. Values may be hard-coded into program code as literals. This saves stack but creates maintenance problems for volatile values.

4. Data values may be loaded into extra data segments. This is especially useful for highly volatile data values in transaction processing applications utilizing multi-threaded process handling. It is a relatively complex approach but can be of great use.

Another available but seldom used technique is to place data values into PB-relative code. The code is SPL and the internal retrieval is via a binary search. This technique has tremendous potential for improving performance.

1. Disc access is eliminated.

2. CPU overhead for retrieval is extremely low.

3. Storage demands are relatively low, particularly if the code resides in readily shareable SL segments.

4. Access is simple since a single call statement is sufficient.

Nothing is free. For all its benefits, PB-relative code has many drawbacks.

1. Dynamic changes cannot be made to PB-relative code.

2. SPL is a requirement for PB-relative binary access.

3. CST table limitations may restrict use of easily shareable SL files.

On balance, I strongly recommend PB-relative techniques in cases where performance potential is needed. I also recommend that they not be applied just because they are available. Like many other techniques, its use must be based upon its relative value within the application..

Two example subprograms are included in this paper. One demonstrates retrieval of fixed length values; the other, variable length. Obviously, either will validate data arguments.

PB-RELATIVE MAINTENANCE (Figure 18)

A major reason to avoid using PB-relative code for validation and conversion purposes is the difficulty it presents to maintenance. The potential benefits cannot, however, be ignored.

Maintenance of tabular data in PB-relative code is not trivial but it need not be excessively difficult. The first step requires that records be prepared for batch input to a maintenance program. This is a common function in all batch maintenance applications and should be no problem.

The actual maintenance run is different depending on how we wish to apply the maintenance. There are three primary approaches to applying maintenance.

1. Maintenance data can be converted to CON (constant) constructs in SPL. These can be inserted into a skeleton program which will be compiled into a USL. This is a straight-forward approach but requires multiple steps.

2. Maintenance data can be applied directly to a generalized USL skeleton. This is quite efficient but requires considerable knowledge of USL structure.

3. Maintenance data can be applied directly to either an RL

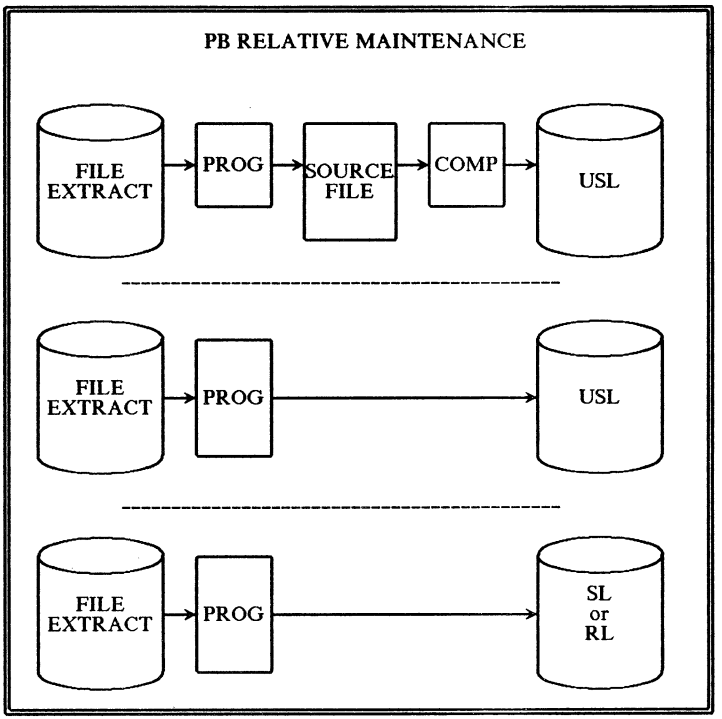


Figure 18

or an SL. This is even more direct than modifying a USL but is probably more difficult.

We have a fourth technique that I hesitated to put on the diagram. We can change the program directly. I left it off the diagram because it destroys my credibility with the fragile types.

Program code is not user-alterable during execution. That does not keep us from altering files just because they happen

to represent programs. There are many things you can do with program files if the need arises. Keep your eyes open for opportunities. It's fun.

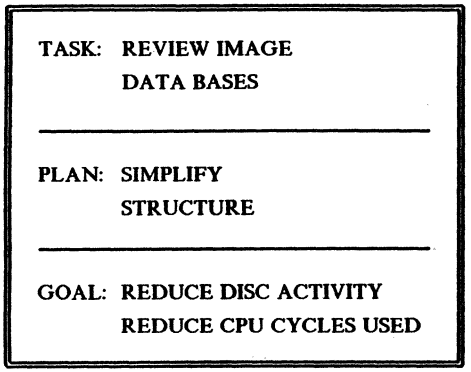


Figure 19

IMAGE DATA BASES (SIMPLIFICATION) (Figure 19)

Commercial applications contain requirements that often lead to complex data structures. IMAGE gives us the means to control these data structures with reasonable ease. Sometimes we forget that what is reasonable for us from a design and access point of view may be totally unreasonable at the machine performance level.

Designing paths to detail data set records is a simple task, particularly if the path is from an automatic master. The desire for rapid random retrieval often justifies this structure.

Using a path from a detail data set to a manual master can provide non-programmatic data validation as well as potentially rapid retrieval. This capability justifies many of our decisions during application and data base design.

When performance becomes unsatisfactory we have to revalue our decisions. With experience, most of us become more selective with the facilities we include in our designs. We have learned that complex, aesthetically pleasing structures may become our white elephants of performance.

TASK: Review IMAGE data base structures.

PLAN: Revalue our design with hopes of simplifying the structure without undue impact on functional performance.

GOAL: Reduce disc activity and CPU load.

REDUCE LINKAGE PATHS (Figure 20)

IMAGE gives excellent performance in random record retrieval. This is one of the major reasons why IMAGE has become so widely accepted. We appreciate the capability to read detail records by multiple key items. Sometimes we get carried away in our appreciation and go too far.

Rapid detail access is achieved using pointers which allow precise record location. These pointers have to be created before the record is accessible. The price of rapid retrieval is paid by the machine when it sets these pointers.

The majority of the overhead for adding a detail record linked to multiple masters comes from establishing the linkages to those masters. On a dedicated Series III you can closely predict that about 7 or 8 linkages can be created or deleted per wall second. This has tremendous implications for performance.

Adding or deleting a detail linked to 7 masters will take about 1 wall second. This is the main reason why IMAGE reloads and batch record maintenance programs run relatively slowly. With 7 linkage paths per detail you can only expect about 4000 adds and deletes per hour.

Transaction processing is also severely impacted by the

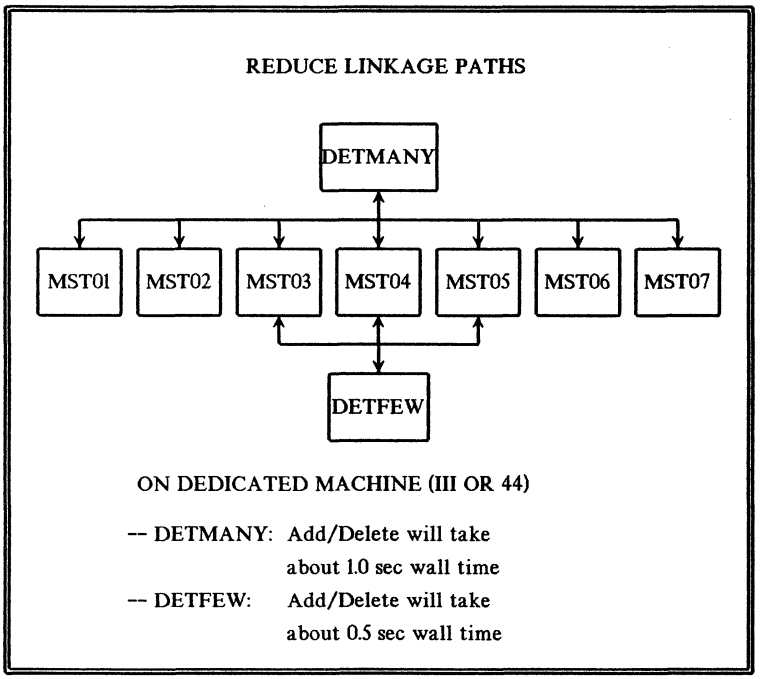


Figure 20

maintenance of linkage paths. If a transaction causes additions of 3 records and each has 7 linkage paths you have every reason to expect at least a 3 second delay during those adds. Under normal circumstances when others are sharing the machine you should not be surprised with much longer delays. This helps explains the following common situation. You can supply the interpretation.

- Design goals call for 5 second maximum processing time delay

- Testing shows excellent average delays of 3.5 seconds

- Initial production delays average a satisfactory 4.5 seconds

- 3 months later the delays are averaging 10 seconds

- You have a problem. The test cases did not predict the realistic performance.

Worst case performance comes during record modification if a search item needs to be changed. This requires a physical delete and add. The time for this change is the sum of the delete and add times. This type of processing can break the back of an application.

All too often the ability to define multiple paths into details seduces us into defining too many paths. Unless the path is required or gives a high priority extra capability you should think seriously before creating it.

1. Every path defined has essentially the same cost.

2. Every path probably does not give the same payback.

3. Is the payback worth the cost?

4. Will the extra path create the monster called "change = delete + add"?

Limiting the number of linkage will improve performance. Consider the case where we go from 7 to 3 linkage paths.

1. Processing delays will drop from 1 to about .5 seconds.

2. Delete and add for change will occur less often and will take only half as long.

3. Delays on a loaded system will probably grow much less and will be less dramatic.

4. Reloads and batch maintenance will be significantly faster.

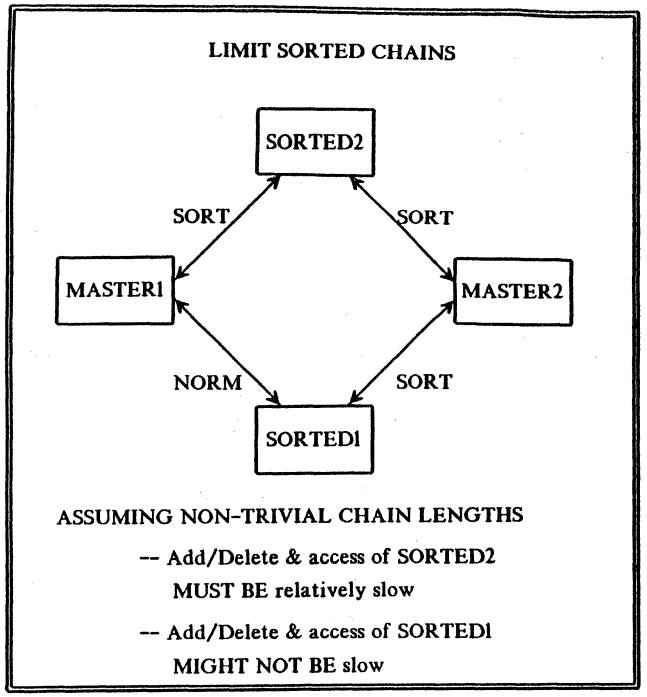


Figure 21

LIMIT SORTED CHAINS (Figure 21)

IMAGE allows details within a logical path to be sorted by some other item value. This is a powerful capability which may be of use to us. It is also an expensive capability.

Addition of a detail with a sorted path causes more overhead because sort sequence must be maintained. IMAGE starts at the logical high end of the chain and follows it backwards until it finds the correct logical home for the detail. It then links the detail into the logical path and goes about its business for the remaining linkage paths.

If the detail fits at the logical end of the chain, the added overhead is trivial. If sort-item values are random, IMAGE must read half through the chain on average to find a logical home. This can be quite expensive.

Note: Sorted paths can give you unpleasant surprises.

1. The value sorted extends from the sort item to the end of the record. This is the "implied sort".

2. Additions will use the entire implied sort value.

3. Any sort before addition should include the full implied sort item. Hideous performance can result if this is not done. This explains why sort items should usually be at the end of a record.

4. Implied sort sequences can be useful. They are also functionally dangerous. IMAGE allows update in place on non-sort items. The implied sequence disappears once updates are done to fields within the implied sort.

Defining multiple sorted paths within a detail guarantees performance degradation. There is no way to avoid extra overhead with multiple sorted paths.

LIMIT LONG OR VOLATILE CHAINS (Figure 22)

The ability to chain logically related records together is a necessary function in any Data Base Management System. This function is designed with relatively short logical chains in mind. It is not meant to be abused.

A good logical chain usually maintains relationships based upon important data items. They are designed to preserve solid logical relationships among records.

1. Customer identities linked to their orders.

2. Line items within an order linked to an order header.

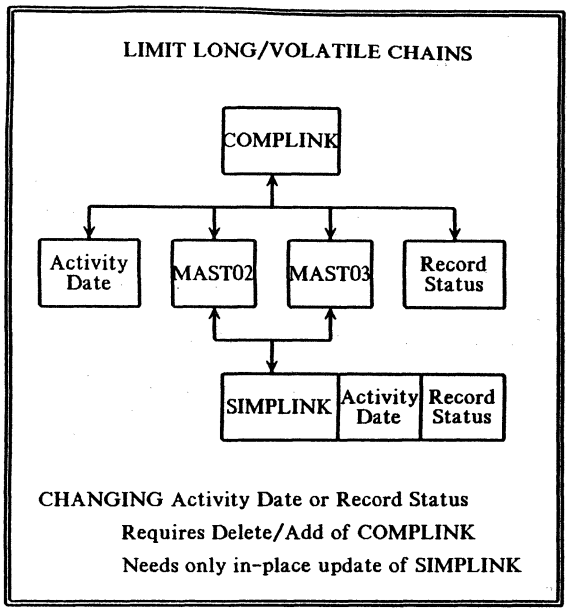


Figure 22

3. Order line items linked to inventory status records.

4. Inventory status records linked to open purchase orders.

5. Purchase orders linked to responsible vendors.

A poor logical chain usually attempts to maintain relationships which are relatively less important. They are often designed to try to create artificial order out of inherent chaos. Even worse, they may exist simply because the capability to create them is provided.

1. Personnel records linked to employee sex.

2. Invoice records linked to invoice status.

3. Student records linked to grade values.

4. Inventory records linked to last activity dates.

Another form of poor chain would be the case where the search value is volatile. Every time the value changes, a complete delete/add must be done. This can be very expensive.

My generalized definition of a poor logical chain is based upon present technology. A poor chain is one whose cost exceeds its value. When technology reduces cost sufficiently, I will change my definition.

The definition of a poor chain is also not absolute. If your application benefits more from a chain than it spends to have the chain, the chain is good. Good and bad are merely a comparison of relative gain to relative cost.

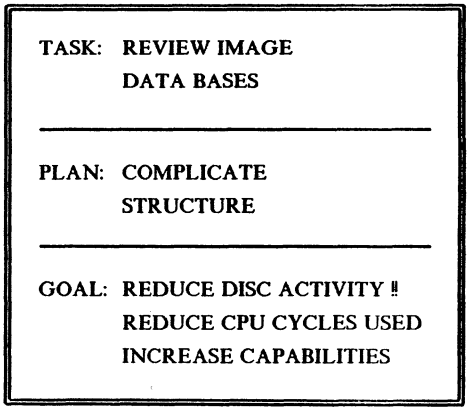


Figure 23

IMAGE DATA BASES (COMPLICATION) (Figure 23)

Simple IMAGE data base structures generally absorb less

overhead per function than do complex structures. This leads us to avoid complex structures to maintain good performance. Sometimes, however, the simple structure becomes a burden and causes unacceptable performance for some required application functions.

Overall performance within an application is a complex entity. Each function performed carries an inherent overhead based upon the function, how often it is performed, and the structure of the data base it accesses. Simple structures can degrade performance if they fail to permit efficient processing of frequently required functions.

Relatively complex structures may benefit overall performance if the structure matches the intended use. Simple structures, while inherently more efficient, may degrade performance if they do not satisfy the application. We cannot judge nor can we design a data base without extensive knowledge of the application.

We may find that our application cannot be serviced satisfactorily without complicating our data base. With proper planning we may be able to use complexity to our advantage to improve both functional and overall performance.

TASK: Review IMAGE data base structures.

PLAN: Revalue our design based upon our knowledge of the application and try to find places where a more complex structure can improve performance.

GOAL: Selectively increase complexity to make our application overhead go down when performing required functions.

SELECTIVE CONSTRUCTIVE ABUSE (Figure 24)

Simple data structures are normally preferable to complex structures. Chains should usually be avoided where data volatility is a problem. But you can throw out any generalized rule if it serves you poorly.

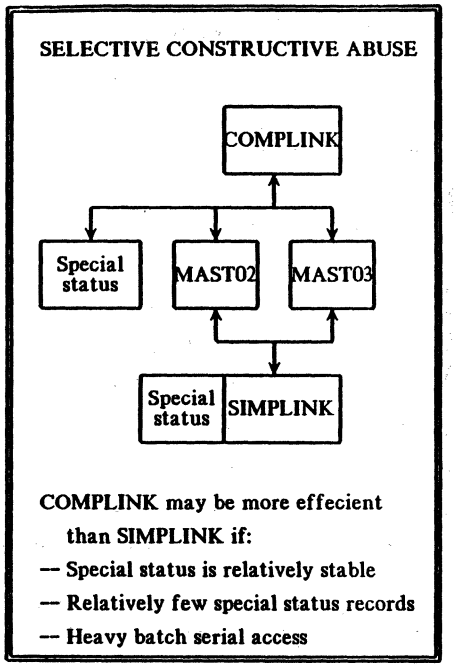


Figure 24

Every application is different. Its success depends much more on how well you have made it perform in the eyes of your users than on how well you have applied standard techniques and principles.

Sometimes you do everything "right" and create a performance dog. You may be stuck in the a lose-lose position.

1. Your application has two obviously good linkage paths. Performance is acceptable for all transaction processing.

2. Nightly batch exception reporting is horrendous. You can't get your special status reports finished by 8:00 A.M.

3. You establish a linkage allowing direct access by status. Note: You're lucky you didn't blow chain length limits.

4. Your nightly exception reporting is now a piece of cake.

5. You have a new problem. Transaction processing is dying. The extra chain and its volatility are eating resources alive.

6. Now what?

The time has come to revalue your position. Your thought and action process might go like this

1. I can only get batch efficiency using the complexity of an added linkage path.

2. I can only get transaction processing efficiency with a simple structure.

3. Maybe if I combine the two structures I can have the best of both worlds. I'll try a more complex physical structure that I can look at from two logical directions.

4. For my normal records I'll have a detail linked by the normal two linkages. As long as the status isn't really special I'll treat it as just another field.

5. For my special records, those with very special status codes, I'll design another detail linked to status.

6. With rare exceptions, my transaction processing will run well just like it did before.

7. My batch exception reports will be printed on time.

8. I think I've got it!

You have abused the data base by making it much more complex. You have been selective in matching your changes to your application. You have been constructive - your application now performs properly.

There are times to follow accepted rules and there are times to write your own rules. The only real problem is knowing when to do which.

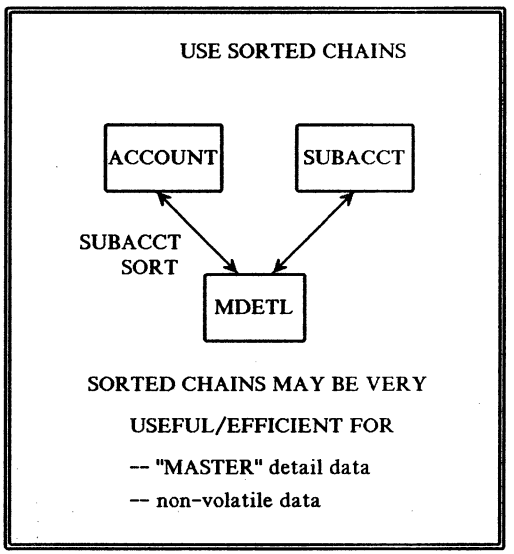


Figure 25

USE SORTED CHAINS (Figure 25)

Sorted chains are powerful tools and carry with them high potential overhead burdens. We should use them carefully but we should not be afraid of them.

Many data base designs could improve functionally if we

designed them with sorted chains. We avoid most potential uses because they will perform poorly. Certain cases will perform extremely well.

1. The detail sets are pure linkage sets: they merely serve to define ordered paths between masters.

2. The detail sets are "master" type details. They represent, for example, what might be considered a master record in an indexed file environment.

3. The data in question is stable.

This structure is extremely useful for keeping master implosion and explosion chains. In an accounting application, they can allow us to explode an account number into sorted sub-account to account.

CONSIDER COMPLEX SORT STRUCTURES (Figure 26)

Sorted chains and complex structures are both potential performance bottlenecks. Functional necessity may require their use.

IMAGE is not designed to give logically sequential access of master data sets. Indexed file structures provide logical sequential access but lack most of the capabilities of IMAGE. We're in that potential lose-lose situation again if we need IMAGE capabilities and have to have logical sequential access.

Some programmers solve this problem by maintaining dual data structures. They create the normal IMAGE data base and keep a separate KSAM file to contain key values extracted from their IMAGE master sets. It works well for most of them.

Other programmers attack the problem by setting out to emulate index type structures within IMAGE. I like this approach because it seems more fun to work with and it keeps everything in IMAGE. Once you define the master you follow this basic plan.

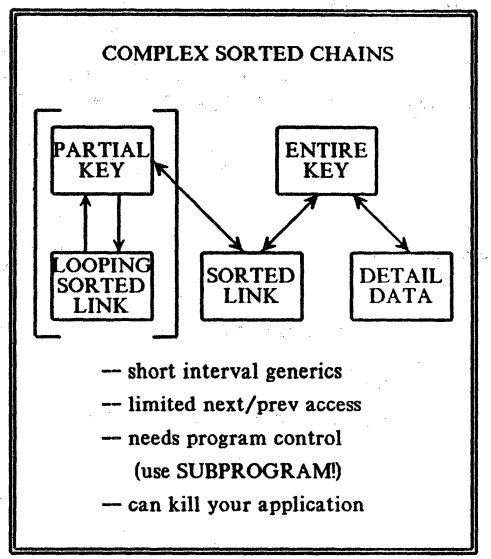


Figure 26

1. Define a path to a detail which links your master to your index emulation sets. This detail is optionally sorted by master value on the linkage path coming from the index emulators.

2. Define a detail index set to hold linkages between partial keys. This set links to your interface linkage detail and has two linkages to a detail linkage set.

3. Define a detail index set to hold linkages between partial keys in the master index set. One path is sorted to point to lower level (longer) partial keys. A second optional sorted path allows reversal in sequence.

4. A partial key of null values starts your emulated index.

5. The rest of the index structure contains gradually longer

partial keys. The common increment is two characters.

Index emulation has both good and bad points. You make your own choices since you pay the bills.

1. Short interval generics become possible.

2. Limited next/previous record capabilities are available. If you try to process the data base purely sequentially you will probably regret your decision.

3. This technique requires extensive programmatic control. I suggest a canned subprogram.

4. You may be tempted too much by logical sequential access. Remember that physical access will actually be totally random and expensive.

5. Your application may die.

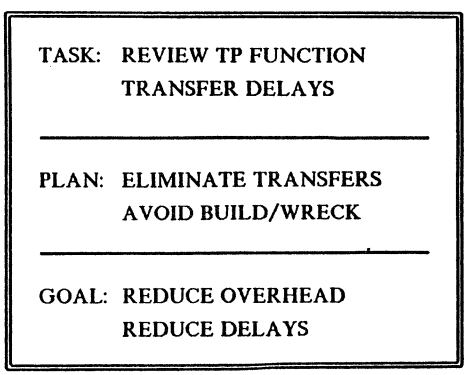


Figure 27

FUNCTION TO FUNCTION TRANSFER DELAYS (Figure 27)

Batch processing applications tend to isolate functionally

similar records into groups (often as physical "batches") and pass them as a unit through a program or program stream. Pure transaction processing applications, on the other hand, provide users the ability to enter virtually any transaction with assurance that processing resources will be available. In theory, no transaction can be predicted before the user presents it to the application.

No matter how well we process a transaction, we are judged harshly if we take too long getting ready to process it. A beautifully performing program gives poor performance if it takes too long to begin executing. Our users will justifiably demand rapid transition between functions.

Transfers from one function to another can also involve considerable overhead. If this overhead is too high the machine can become so bogged down it has little left for its true job, processing transactions. Successful applications spend their resources processing, not getting ready to process.

We usually test programs for their ability to process transactions efficiently. We often forget that much more than efficient programs is needed for effective applications. We have to be concerned with function to function transfers if we expect success.

TASK: Review function-to-function transfers.

PLAN: Try to eliminate transfers if possible. When transfers cannot be eliminated, try to reduce their cost by reducing their overhead.

GOAL: Reduce system overhead in general. Reduce transfer delays to improve user perception of performance.

MAGNIFICENT MASOCHISM (Figure 28)

Transaction processing applications are often complex. Demands for extreme flexibility stretch programmers skills to the limit. Process handling is an easily implemented

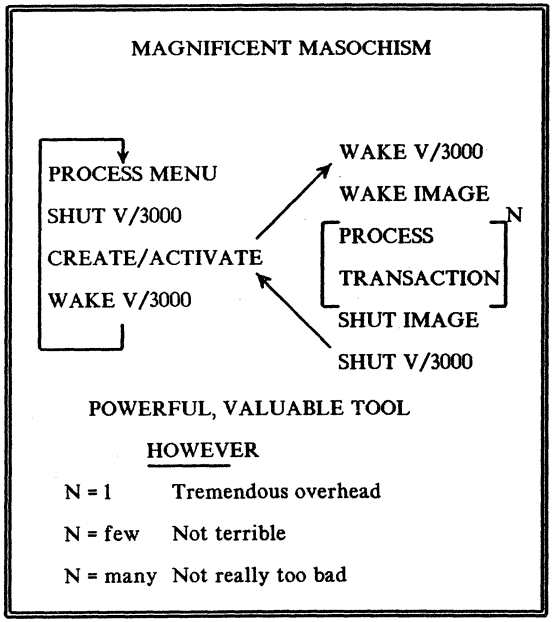


Figure 28

technique often used to help simplify a complex situation. It can be quite effective in the proper circumstances and can beat the machine senseless when misused.

Most users have plenty to keep them busy. They justifiably require the simplest possible interface between themselves and the machine. Most commercial software packages are menu driven to help win acceptance in the market.

Menu driven applications are easy to design and document. They can also be reasonably simple to program and test. Their biggest potential problem is a tendency toward tedious maintenance and relatively poor performance.

Modular programming techniques reduce the difficulty of

maintenance. The most modular technique involves isolating individual or similar functions in separate programs which can be controlled using process handling. A selection of a menu item triggers a programmatic execution of the appropriate program. In effect, the menu program issues predefined "run" commands which are functionally invisible to the user.

Function invisibility is not true invisibility. The user sees the execution of the internal "run" as a delay in processing. There is considerable overhead associated with both starting and stopping a program. The overhead to execute a normal transaction is usually much less than the overhead spent invoking its program.

When the selected program will execute numerous transactions before returning to the menu the overhead burden may be quite acceptable. As the number of transactions per execution gets smaller, the overhead becomes objectionable. If the design calls for only 1 transaction per execution the application is potentially terminally ill.

A standard technique to allow flexibility without tremendous overhead is to keep a program alive after it has been invoked. Process handling allows reactivation of a suspended program (process) with minimum overhead and delay. V/3000 processing may create some problems, however, and the limit on the number of processes alive within MPE may curtail your ability to use this technique.

VIABLE ALTERNATIVE (Figure 29)

Some programs promise so much power and flexibility that it is impossible to avoid some performance loss. With intelligent programming we can minimize the loss.

Subprograms allow almost as much flexibility as process handling and usually require less overhead. Critical values such as IMAGE and V/3000 control areas can be passed as parameters from caller to callee. A standard technique involves defining a single data area containing the most

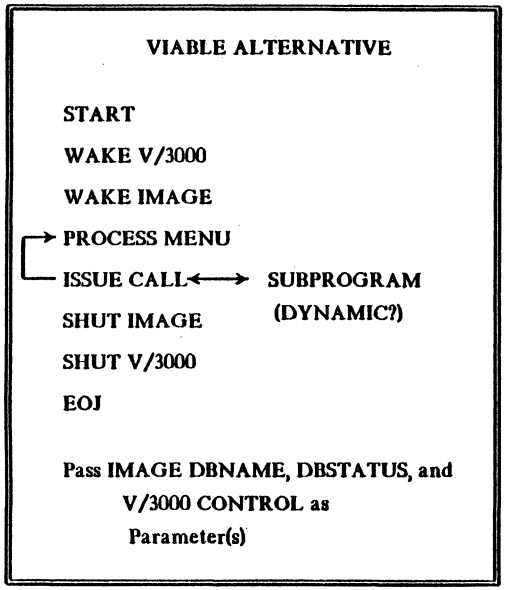


Figure 29

common data and passing it to all subprograms as a single parameter.

Stack sizes may be a limit on the use of subprograms. Dynamic subprograms can help control stack size but must be used carefully to avoid excessive overhead of their own. Excessively large stacks inhibit performance and so can excessive stack expansion, contraction, and initialization.

Dynamic subprograms also limit capabilities of the subprogram. Values kept in local storage is lost when you exit the subprogram. If files are opened they must be closed before you exit. More care is needed when you decide to make an existing subprogram dynamic than when you originally design it to be dynamic.

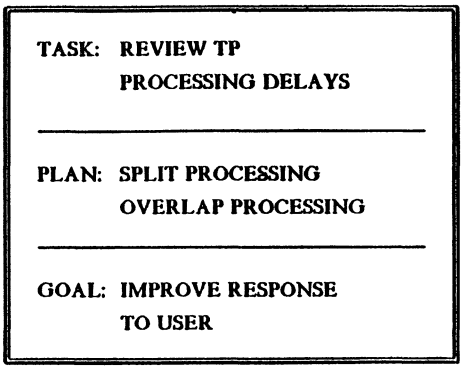


Figure 30

PROCESSING DELAYS (Figure 30)

Processing the actual transaction data is the eventual goal of transaction processing. The time taken for processing ranges from a very small percentage of the total transaction time to a very large percentage. When the percentage is too large we have to try to reduce it to a more acceptable level.

Processing normally breaks down into two phases, data validation and data storage or retrieval. Either or both phases may require extensive disc access with corresponding overhead.

Assuming overhead has already been minimized, there is virtually no way to do anything with the data validation phase. If we expect to reduce processing delays, we can make improvements only in the storage or retrieval phase.

MPE contains easily implemented techniques that allow us to improve performance as perceived by the user. Improvement in perceived performance will not reduce machine overhead and may in fact increase it. In the final analysis, though, increased machine overhead is meaningless so long as the user, the authoritative judge of performance, sees

improvement.

TASK: Review processing delays.

PLAN: Evaluate data storage and retrieval delays. When excessive, attempt to use MPE capabilities to reduce or eliminate them.

GOAL: Improve perceived performance as measured by the user.

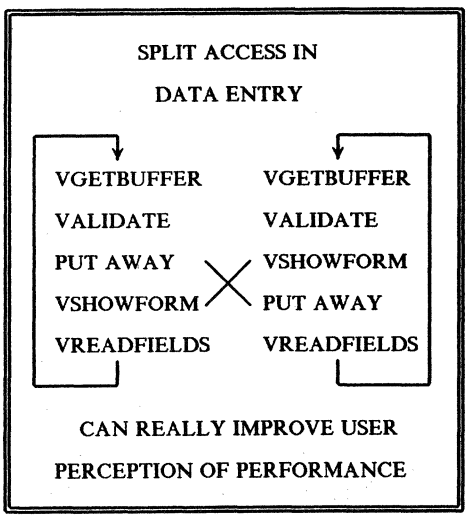


Figure 31

SPLIT ACCESS IN DATA ENTRY (Figure 31)

Performance as perceived by the user is or probably should be our primary concern in transaction processing. This is most critical in data entry applications where uniform response helps set up a work rhythm.

Data entry may involve one or many screens per logical transaction. Multi-screen transactions usually benefit from hardware advances such as the ability to use screens downloaded to the terminal. Single screen transaction performance can be influenced by programming.

Screen data is usually validated and then stored on disc. When the storage is in a data base the overhead for storage may take much time. This time is usually the major contributor to delays in processing functions in data entry.

V/3000 can be used to reduce the delays in data entry. Once you have validated the screen and are certain that only a catastrophic failure could keep you from storing the record, you can return the screen to the user for entry of the next record. While the user fills the next screen, you put away your record.

No data will be lost if the record is not stored before the user hits the enter key or a soft key. These attempted transmissions will be ignored until the program issues a read. In all but a few special cases the validated record will be stored well before the user has submitted the next screen.

This technique will not reduce overhead or performance of individual functions within the program. Responsiveness to the user, however, will be improved. The degree of improvement is proportionate to the amount of work being done parallel to the next transaction think time. An additional benefit is that response will become somewhat less dependant upon overall system load.

PROCESS HANDLING IN DATA INQUIRY (Figure 32)

Application programs often have only the simple capability to allow user data inquiry. These programs must perform well but users expect and usually tolerate reasonable response delays. When the data inquiry is part of a larger function those delays must be minimized.

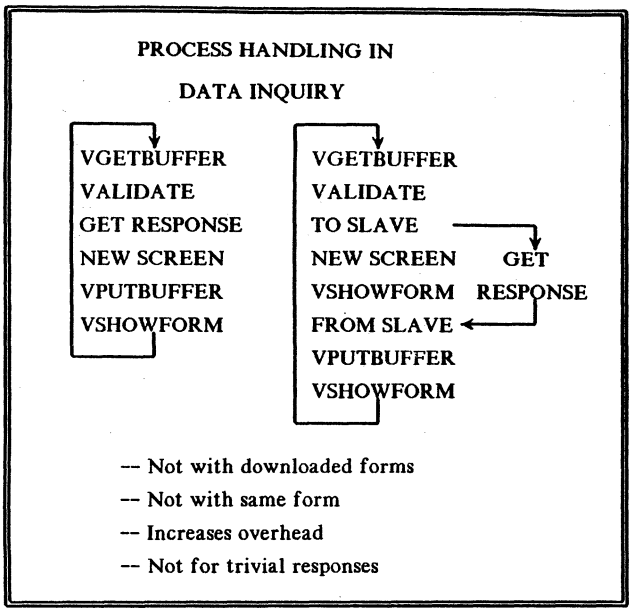


Figure 32

Programs which update existing records have characteristics of both data entry and inquiry. The inquiry part can be a major performance bottleneck. This is quite common since update programs often must retrieve multiple records even though only one may actually be subject to change.

Additional delays are built into such programs if the modifiable response must be displayed on a new screen. User patience is severely tested in many applications because we program serially.

1. We read the request and validate its content. We hit the screen fast if we find errors.

2. We are not so friendly with good input. The user must

wait for us to access our data bases to build a reply. Their reward for good input is the chance to watch a cursor blink.

3. We finally respond by sending out a new screen and allow user modification.

4. We satisfy user function but we do it in slow, easy steps.

Process handling can be used to make a program both more responsive and more friendly. We can perform critical time consuming steps in parallel so that delays visible to the program can be invisible to the user.

1. We read requests and handle errors as usual.

2. For good input we immediately send a "please retrieve this data" message to a slave process we have created. The slave was waiting patiently since its only job is to retrieve records from data bases.

3. While the slave gathers records we paint the response screen. This is a friendly act. Our response to good input is probably only a fraction of a second slower than a response to bad input.

4. The faster parallel function waits for the slower to finish. We then retrieve the response buffer from the slave and write it to the screen.

5. Except for a short time needed for process to process communication we have reduced user delay to the delay of the longer parallel process. Even more important, we have become responsive to our users.

This technique is valuable because it improves performance within a technologically limited environment. Changes in technology could make it less valuable or even useless. For example, this technique has no benefit when screens have been downloaded to the terminal since that technology has already eliminated screen painting delays.

We also would not use this technique for trivial responses which take little time to prepare. Process handling absorbs overheads and may not always be cost effective.

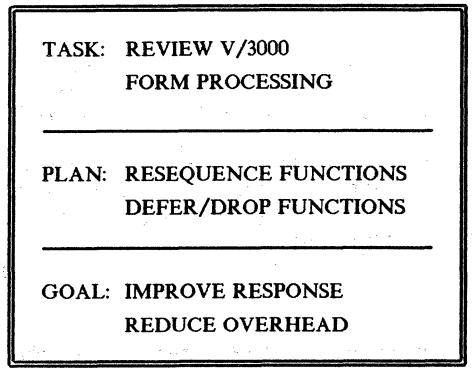


Figure 33

V/3000 FUNCTIONAL SEQUENCING (Figure 33)

Your application defines the V/3000 screens you and your users create for the person to machine interface. Although physical screen design may be critically important to performance it is too broad a subject for this paper. We'll have to assume you've already designed satisfactory screens.

After the screen has been filled, the programmer can begin to manage the processing of the input. Application requirements and V/3000 protocols must be satisfied but the programmer has many options. Some of these options can greatly influence performance.

Application requirements must be met. The programmer should interpret these requirements, however, to see if they can be resequenced more efficiently in the program. Resequencing internal events often changes performance.

V/3000 protocols must also be satisfied. The V/3000 documentation defines a hierarchy among the V/3000 intrinsics and among the functions performed for us by the V/3000 edits. That hierarchy is also open to programmer interpretation. We may be able to improve performance by changing our use of the V/3000 intrinsics.

TASK: Review V/3000 form processing

PLAN: Resequence functions selectively, defer functions when practical, drop functions where possible

GOAL: Improve performance within the program and as perceived by the user.

SOFT KEYS AND SELECTIVE EDITING (Figure 34)

Dumb terminals and unsophisticated terminal I-O limit transaction processing. Intelligent terminals and more sophisticated terminal I-O interfaces remove many of these limitations. We may also have to become more intelligent and sophisticated to more fully utilize our better tools.

Soft keys were among the first improvements as terminals began evolving from absolute dumb to somewhat smart. We take them for granted and use them for standard functions such as "exit" or "refresh". Few of us are using them fully. This paper cannot attempt to cover this topic but we can justify looking at a frequently overlooked usage.

Many applications could flow more smoothly if soft keys could be used to trigger processing functions and if screen information could also be available for processing. A common design technique is to require the user to hit the function defining soft key and then hit "enter" to transmit the buffer. This works but it is neither sophisticated nor friendly.

V/3000 can work with the terminal to allow reading the screen after a soft key. We can trigger this function, called auto-read, any time we wish using a simple subprogram or any other

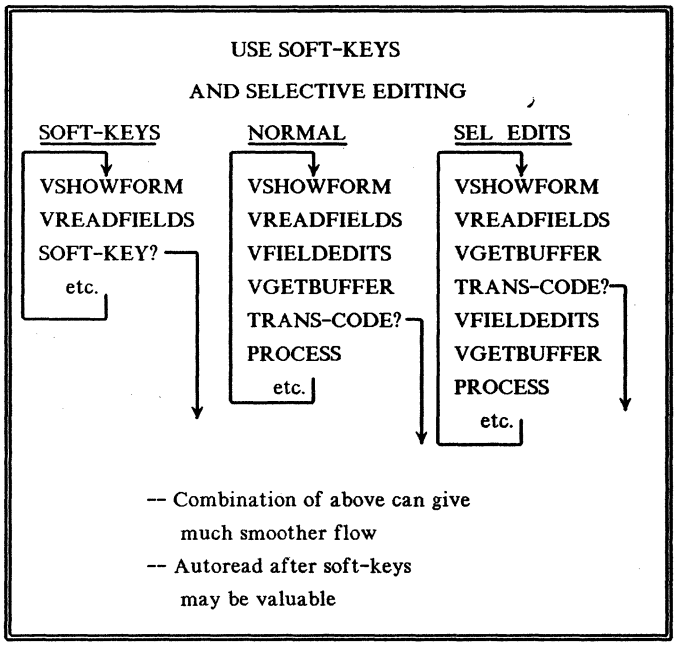


Figure 34

technique to set the autoread bit in the V/3000 common area. Autoread can help make an application run more smoothly and be more friendly. That qualifies as a performance improvement.

Another facility we often ignore is the ability to edit data selectively. V/3000 documentation implies that screen edits must preceed program edits. This is a valid standard for most cases but it may not fit comfortably into all situations. Fortunately, the implied sequence is not mandatory.

When we assume that all screen edits must be done before any program edits we may be painting ourselves into a corner.

Why, for example, should we edit an order quantity when the ordered item is an invalid product. A standard workaround is to avoid V/3000 edits in favor of program edits. Like many workarounds, it works but denies us access to a useful subsystem capability.

We can sometimes "have our cake and eat it too" if we become more flexible. Consider the following sequence of events that uses full V/3000 field edits but allows program intervention at a critical point. This is only a simplified example of sequence editing.

1. Read the screen normally using VSHOWFORM and VREADFIELDS

2. Don't edit the screen yet. Issue a call to VGETBUFFER so you can check a critical field in your program

3. If the critical field fails a test, give the screen back with an appropriate message

4. If the critical field is acceptable, go back to "normal" processing. Issue calls to VFIELDEDITS, etc.

5. You have it made. You get program control at a critical stage and you use V/3000 for less critical work.

DEFER OR DROP PROCESSING (Figure 35)

V/3000 gives us much power with little programming effort. When used sensibly, V/3000 can be good for both the programmer and the user. When used only from the programmers point of view, V/3000 can be a nuisance to the user.

V/3000 gives us three standard processing phases. The initialization phase is relatively little used and causes few problems. We seldom use the edit and finish phases absolutely wrong but we often cause excessive overhead and user aggrevation.

Unless there is some special requirements in an application, some edit functions should never be considered in the edit

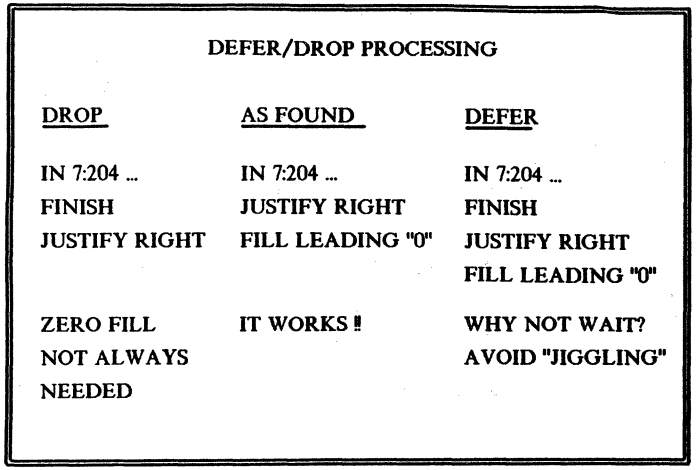


Figure 35

phase. Whenever we justify or fill a field in an edit phase statement we are performing a potentially wasted function. Even more important, why should we run the risk of having to rewrite a valid field just because we have altered its form? Users object to excessive screen "jiggling".

We have done only part of our job when we isolate functions within logical V/3000 phases. Whenever possible we should avoid the finish phase until we fully accepted the screen. We waste overhead and risk "jiggling" when we perform this work prior to full screen acceptance.

Some functions can be dropped entirely. Until the machine or operating system are changed, we do not have to fill leading blanks with zeros in numeric fields. The machine treats leading blanks as zeros and allows us to drop the fill function if absolutely necessary for performance.

INTRINSICS AND LANGUAGES (Figure 36)

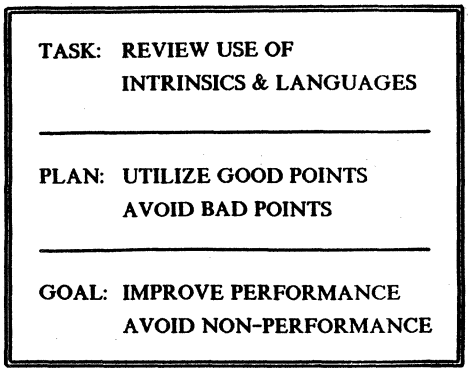


Figure 36

Most programmers prefer a particular language and are proficient in it. In the average application the choice of languages will not determine program performance. We cannot guarantee that any one language is inherently better than another in all possible applications.

We also know that the average application often requires specific functions that may not be available in our language of choice. If we work in a multi-lingual shop we may be able to have a special subprogram written in a language that supplies that function. If our shop is single language we either avoid special functions or devise emulation techniques of some sort.

MPE assists all of us by providing intrinsics for many special functions. Many functions not built into a compiler may be available through an intrinsic. This can simplify our programming and make our jobs much easier.

Somehow we will find ways to get our programs running. All too often we immediately have to find a way to get them running more efficiently. Part of that may require a revaluation of how we have used our languages and the intrinsics.

TASK: Reviews use of languages and intrinsics

PLAN: Use strengths of each and try to avoid weaknesses

GOAL: Improve performance. Of utmost importance, avoid non-performance.

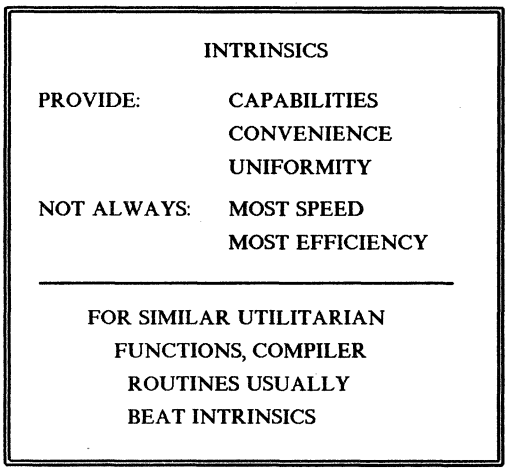


Figure 37

INTRINSICS (Figure 37)

Intrinsics are powerful generalized routines provided with MPE. Some allow limited access to highly specialized functions and others give standardized access to common utility functions.

Intrinsics offer the only access to many functions not available unless you program at the machine instruction level and execute in privileged mode. Access to I-O, for example, is only available through the intrinsics. Compiler library modules invoked by the compilers eventually call intrinsics

for all file access.

All "special capabilities" are available only through the intrinsics. Process handling and data management are invoked using a relatively small subset of the intrinsics. Programmatic communication between computers is handled in similar fashion.

Intrinsics also provide convenient access to many utilitarian functions. Binary to Ascii and Ascii to Binary conversions are common examples. Serial table searches and character transformations are also available.

Specific parameter format and sequence requirements guarantee uniformity. Except for individual compiler conventions and limitations, once you learn to use an intrinsic in one language you should be able to use it in all languages.

SPL comes closest to accessing intrinsics in strict conformance with their documentation. All other languages provide higher level interfaces to one degree or another.

Intrinsics are carefully coded and function efficiently. They are also highly generalized. This generalization leads to relatively high internal overhead. The intrinsic may expend a good portion of its efforts isolating the particular subfunction it is being asked to perform.

Intrinsics will generally be marginally more efficient for file manipulation than the corresponding compiler modules. The difference is slight and seldom justifies the required attention to minute detail. This generalization is meaningless when we use the intrinsics to reach capabilities not available within a particular language.

Compiler routines usually handle discrete data manipulations more efficiently than would the corresponding intrinsic. Even when the compiler routine internally invokes an intrinsic it will not be meaningfully degraded.

Arithmetic functions are particularly well handled by compiler routines. Many of these routines generate highly

efficient machine code. The COBOL compiler uses packed decimal instructions extensively and is surprisingly efficient in handling numeric data.

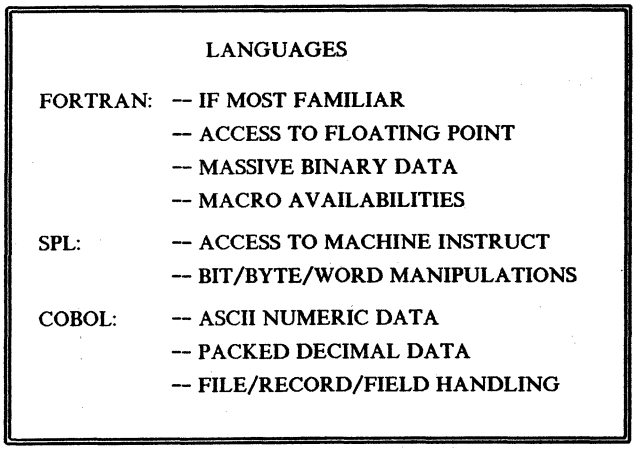


Figure 38

LANGUAGES (Figure 38)

No language is perfect. Each has its special strengths and weaknesses. A language should be evaluated according to how it fits into the application environment.

FORTRAN is an established language with many strong supporters. Programmers either love it or hate it. Either way, there are times when it should probably be your language of choice.

1. If FORTRAN is most familiar to you, why not use it? Most of us usually produce better work when we work with a known quantity.

2. FORTRAN offers excellent high-level access to floating point arithmetic. This is a major strength of FORTRAN and often virtually necessitates its use. FORTRAN is often used to write subprograms accessible from other languages.

3. FORTRAN works very close to the machine when doing binary arithmetic. Massive calculations with binary data often justify FORTRAN for performance reasons.

4. FORTRAN includes many high-level macro constructs. These may make FORTRAN more useful than other languages.

5. Unfortunately, FORTRAN performs poorly with Ascii numeric data and in its current implementation cannot handle packed decimal numerics. Since these data formats are used extensively in commercial applications, FORTRAN is usually a poor choice there.

SPL is the lowest level language on the HP3000. As such, it is potentially the most efficient. It is also relatively tedious compared to most high-level languages.

1. Although SPL is potentially more efficient than any other languages it will probably not improve performance enough to justify wholesale use in commercial applications. Higher level languages are simply better suited for general purpose use.

2. SPL usually fits in best when used for writing specialized subprograms. Some functions are simply not handled well by high-level languages.

3. Machine instructions can be reached in SPL. This is particularly valuable for string and bit manipulations.

4. SPL is most suited for work with low level data. It works especially well at the word level and below. SPL handles records and fields well but the source code tends to become cumbersome and difficult to maintain in large programs.

COBOL is the most widely used high-level commercial language.

Most programmers hate COBOL, some grudingly accept it, and almost none will publicly admit a preference for it. For all its faults it remains the language of choice in most shops.

1. COBOL has a reputation as an inefficient language. COBOL is definitely not the most efficient language but its wide use implies acceptable performance. There is little question that it is highly effective.

2. COBOL is quite efficient in handling numeric Ascii data. Ascii numeric functions are performed using packed decimal arithmetic. Conversions between Ascii and packed decimal data are done efficiently by native machine instructions.

3. Packed decimal data is a standard data format in COBOL under MPE. The ability to deal with packed numerics is a definite advantage for conversions or interfaces with the best known Brand-X computer.

4. COBOL is a high-level language designed to be effective with high-level data. It is most powerful when used to process files, records within files, and fields within records. This is probably the main reason for its wide acceptance.

SECTION 3 MIXED BAG

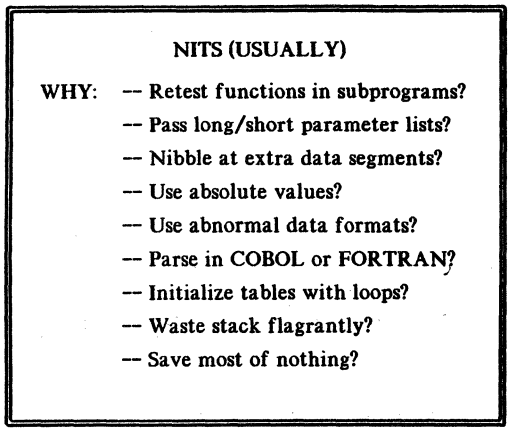


Figure 39

NITS (USUALLY) (Figure 39)

We concern ourselves too much with minor programming considerations. They usually influence performance only slightly. That is not justification, however, for us to code inefficiently.

1. Unless we are writing highly generalized subprograms we probably waste overhead retesting functions. We know why we call a subprogram. In most cases we would be more efficient by writing subprograms with multiple entry points. This avoids retesting to identify our request and also results in more readable programs.

2. Many words have been written about the effect of parameter lists on subprogram efficiency. Long parameter lists are less efficient than short ones. On the other hand, artifically short lists are eventually less efficient if we waste CPU power loading and unloading common control areas. How we enter a subprogram is probably meaningless unless we are doing very little work within the subprogram.

3. Extra data segements have many uses. They can be used quite efficiently but not without added overhead. Whenever data must be moved to or from an extra data segment we should move as much as practical per access. Moving a single word twice costs about the same as moving over 1000 words once.

4. Absolute numeric values are often needed. They cause extra overhead, however, when used in calculations, especially when the result of the calculation is an absolute value. Absolute values force the compiler to generate extra code to guarantee proper results.

5. Packed decimal data has a natural format containing an odd number of numeric digits, each taking up one 4 bit nibble. These plus a 4 bit numeric sign fills complete bytes. If you specify an even number of decimal digits for packed data you force the compiler to do extra work controlling the low order nibble.

6. Character strings are best parsed by specialized routines which utilize special machine instructions. These routines may be designed directly into a compiler or are easily written in SPL. Complex byte manipulations and loop constructs written in highlevel languages are inefficient and should be avoided in most cases.

7. Tables are often initialized using a program loop which

indexes through the table depositing values along the way. This is relatively inefficient and cumbersome. You can save overhead by filling only the first entry and then performing an overlapping move into the rest of the table.

8. Stack space is valuable and should be used with care. Program-directed literals save the stack area needed for valued data elements. But programmers often value documentation over stack. Even so, using a 132 character data element full of spaces to clear a print record is intolerable flagrant waste.

9. Programmers should be efficient but they have to be reasonable first. There is no way we can justify spending time optimizing a small inefficiency when the same effort could be more productive elsewhere. Inefficient coding in insignificant routines does not make inefficient programs.

DON'T ... BUT ...

DON'T	Believe everything
BUT	Believe something
DON'T	Challenge everything
BUT	Challenge something
DON'T	Optimize everything
BUT	Optimize something
DON'T	Quantify everything
BUT	Quantify something

Figure 40

DON'T ... BUT ... (Figure 40)

What was impossible or ridiculous in the past may be standard practice today. What is absolutely true today will quite often be made false by the technology of tomorrow. There are no absolutes but there may be some valuable guidelines.

I'm a programmer, not a philosopher. In Figure 40, I put some comments that have a certain meaning to me. I think they will be more meaningful to you if you supply your own interpretations and meanings.

Good luck and good programming.

SECTION 4 SAMPLE SOURCE LISTINGS

```
* ******************************************************
* THIS IS THE MASTER PROGRAM USED AS A DRIVER TO SHOW THE
*  TECHNIQUES OF HAVING SORTS EXECUTE IN A SLAVE PROGRAM
* THIS PROGRAM COULD HAVE DRIVEN MULTIPLE SLAVE SORT
*  PROGRAMS AT THE SAME TIME IF I WANTED TO TAKE UP THAT
*  MUCH CODE SPACE IN THE HANDOUT
* ******************************************************

$CONTROL USLINIT
 IDENTIFICATION DIVISION.
* DRIVER TO TEST PROGRAM SORTS1
 PROGRAM-ID. FREPSRSS.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. X.
 OBJECT-COMPUTER. Y.
 SPECIAL-NAMES.
     CONDITION-CODE IS COND-CODE.
 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  UNBLOCKER              PIC S9999 COMP.
 01  SAVED                  PIC S9999 COMP.
 01  PROG                   PIC X(10) VALUE "SORTS1R".
 01  PIN                    PIC S9999 COMP.
 01  BFACT                  PIC S9999 COMP VALUE 13.
 01  DSEG-IND               PIC S9999 COMP.
 01  DSEG-ID                PIC S9999 COMP VALUE 43.
 01  DSEG-LEN               PIC S9999 COMP VALUE 79.
 01  REC-AREA.
     05  CURR-COUNT         PIC S9999 COMP VALUE 0.
     05  EACH-REC           PIC X(12) OCCURS 13 TIMES.
 01  DATA-REC.
     05  SEND-COUNT         PIC 999 VALUE 55.
     05  SEND-PROC          PIC 9(9).
 PROCEDURE DIVISION.
```



```
START-ITS SECTION.
START-IT.
    CALL INTRINSIC "GETDSEG" USING
        DSEG-IND DSEG-LEN DSEG-ID
    IF COND-CODE < 0
        CALL INTRINSIC "QUIT" USING 101
    .
    CALL INTRINSIC "CREATE" USING
        PROG \\ PIN
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 201
    .
    PERFORM SEND-EMS UNTIL
        SEND-COUNT < 1
    MOVE -1 TO CURR-COUNT
    PERFORM ACTUAL-SENDS
    MOVE 0 TO CURR-COUNT
    PERFORM GET-EMS UNTIL
        CURR-COUNT = -1
    DISPLAY "THAT'S ALL, FOLKS"
    STOP RUN
    .
GET-EMS SECTION.
GET-EM.
    CALL INTRINSIC "DMOVIN" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 204
    .
    MOVE 1 TO UNBLOCKER
    PERFORM DOITS UNTIL
        UNBLOCKER > CURR-COUNT
    IF CURR-COUNT NOT = -1
        CALL INTRINSIC "ACTIVATE" USING
            PIN 3
        IF COND-CODE NOT = 0
```

```
            CALL INTRINSIC "QUIT" USING 205
    .
DOITS SECTION.
DOIT.
    DISPLAY "FROM SORT " EACH-REC(UNBLOCKER)
    ADD 1 TO UNBLOCKER
    .
SEND-EMS SECTION.
SEND-EM.
    MOVE 0 TO CURR-COUNT
    PERFORM LOAD-BUFFERS UNTIL
        CURR-COUNT = BFACT OR SEND-COUNT < 1
    PERFORM ACTUAL-SENDS
    .
ACTUAL-SENDS SECTION.
ACTUAL-SEND.
    CALL INTRINSIC "DMOVOUT" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 303
    .
    CALL INTRINSIC "ACTIVATE" USING PIN 3
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 304
    .
    MOVE 0 TO CURR-COUNT
    .
LOAD-BUFFERS SECTION.
LOAD-BUFFER.
    SUBTRACT 1 FROM SEND-COUNT
    CALL INTRINSIC "PROCTIME" GIVING SEND-PROC
* IF YOU RUN THIS TEST VERSION, DON'T ASSUME COBOL IS
*  SLOW JUST BECAUSE THE CPU TIME SHOWS 4-5 MILLISECOND
*   BETWEEN RECORDS.  THAT TIME PRIMARILY REPRESENTS
*    THE CPU TIME NEEDED TO DO THE DISPLAY STATEMENT
    ADD 1 TO CURR-COUNT
    MOVE DATA-REC TO EACH-REC(CURR-COUNT)
    DISPLAY "TO SORT " DATA-REC

* ************************************************************
* ************************************************************
```



```
* ************************************************************
* THIS IS A SAMPLE OF A SLAVE SORT PROGRAM CONTROLLED BY
*  PROCESS HANDLING TO ALLOW SORTING OF RECORDS OUTSIDE
*  THE MASTER PROGRAM
* MULTIPLE SUCH PROGRAMS CAN BE CONTROLLED AT THE SAME
*  TIME BY THE MASTER
* ************************************************************
$CONTROL USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS SORTS1S, PROGRAM KEPT AS SORTS1R
*  RECEIVES RECORDS FROM SORTMASR
*  SORTS
*  RETURNS SORTED RECORDS TO SORTMASR
 PROGRAM-ID. FREPSRTS.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. X.
 OBJECT-COMPUTER. Y.
 SPECIAL-NAMES.
     CONDITION-CODE IS COND-CODE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SORTFILE ASSIGN TO "TEMPSORT,,,,100".
 DATA DIVISION.
 FILE SECTION.
 SD  SORTFILE.
 01  SORTREC.
     05  KEY1              PIC 999.
     05  FILLER            PIC X(7).
     05  KEY2              PIC 99.
 WORKING-STORAGE SECTION.
 01  SORT-FLAG             PIC XX VALUE LOW-VALUES.
 01  TO-WAIT               PIC S9999 COMP VALUE 3.
 01  BFACT                 PIC S9999 COMP VALUE 13.
 01  DSEG-IND              PIC S9999 COMP.
 01  DSEG-ID               PIC S9999 COMP VALUE 43.
 01  DSEG-LEN              PIC S9999 COMP VALUE 79.
 01  REC-AREA.
     05  CURR-COUNT        PIC S9999 COMP VALUE 0.
     05  EACH-REC          PIC X(12) OCCURS 13 TIMES.
 PROCEDURE DIVISION.
 START-ITS SECTION.
```

```
START-IT.
    CALL INTRINSIC "GETDSEG" USING
        DSEG-IND DSEG-LEN DSEG-ID
    IF COND-CODE < 0
        CALL INTRINSIC "QUIT" USING 101
    .
    SORT SORTFILE
        ASCENDING KEY KEY2
        DESCENDING KEY KEY1
        INPUT PROCEDURE GET-INS
        OUTPUT PROCEDURE SEND-BACKS
    STOP RUN
    .
SEND-BACKS SECTION.
SEND-BACK.
    MOVE LOW-VALUES TO SORT-FLAG
    MOVE 0 TO CURR-COUNT
    PERFORM GET-AND-SENDS UNTIL
        SORT-FLAG = HIGH-VALUES
    PERFORM SEND-DATAS
    MOVE 0 TO TO-WAIT
    MOVE -1 TO CURR-COUNT
    PERFORM SEND-DATAS
    .
SEND-DATAS SECTION.
SENDING.
    CALL INTRINSIC "DMOVOUT" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 102
    .
    CALL INTRINSIC "ACTIVATE" USING 0 TO-WAIT
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 103
    .
    MOVE 0 TO CURR-COUNT
    .
GET-AND-SENDS SECTION.
GET-AND-SEND.
    IF CURR-COUNT NOT < BFACT
        PERFORM SEND-DATAS
```



```
    RETURN SORTFILE AT END
        MOVE HIGH-VALUES TO SORT-FLAG
    .
    IF SORT-FLAG NOT = HIGH-VALUES
        ADD 1 TO CURR-COUNT
        DISPLAY "FROM SORT IN SORTS1R " SORTREC
        MOVE SORTREC TO EACH-REC(CURR-COUNT)
    .
GET-INS SECTION.
GET-IN.
    PERFORM GET-IN-LOOPS UNTIL
        CURR-COUNT = -1
    .
GET-IN-LOOPS SECTION.
GET-IN-LOOP.
    CALL INTRINSIC "DMOVIN" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 111
        STOP RUN
    .
    PERFORM UNLOAD-EMS UNTIL
        CURR-COUNT < 1
    IF CURR-COUNT NOT = -1
        CALL INTRINSIC "ACTIVATE" USING 0 TO-WAIT
        IF COND-CODE NOT = 0
            CALL INTRINSIC "QUIT" USING 105
    .
UNLOAD-EMS SECTION.
UNLOAD-EM.
    RELEASE SORTREC FROM EACH-REC(CURR-COUNT)
    DISPLAY "TO SORT IN SORTS1R " EACH-REC(CURR-COUNT)
    SUBTRACT 1 FROM CURR-COUNT

* ************************************************************
* ************************************************************


* ************************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES ASCII NUMERICS TO A COBOL SUBPROGRAM AND A
```

```
*  FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
* THE SUBPROGRAMS EACH ADD 100 6 DIGIT ASCII NUMBERS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK    16635 CPU MILLISECONDS
*    FORTRAN TOOK 76217 CPU MILLISECONDS
* ********************************************************

$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS FORCOBDA
 PROGRAM-ID. TESTFORD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                  PIC S9(9) COMP.
 01  STAMPE                  PIC S9(9) COMP.
 01  STAMPD                  PIC 9(6).
 01  THE-SUM                 PIC 9(8).
 01  THE-SUMD                PIC 9(8).
 01  THE-COUNT               PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM             PIC 9(6) OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 6 DIGIT ASCII ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING @THE-TABLE @THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
```



```
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM FOR-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
     STOP RUN
     .
 LOAD-EM.
     ADD 5 TO THE-COUNT
     MOVE THE-COUNT TO THE-NUM(THE-IND)
     .
 COB-SHOT.
     CALL "COBADD" USING THE-TABLE THE-SUM
     .
 FOR-SHOT.
     CALL "FORADD" USING @THE-TABLE @THE-SUM
     .
* ********************************************************
* ********************************************************


* ********************************************************
* THIS IS THE COBOL SUBPROGRAM MENTIONED IN THE PRECEEDING
*  COBOL MAIN PROGRAM. IT IS AN ASCII NUMBER CRUNCHER
* ********************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBSA
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT               PIC S9(9) COMP-3.
 LINKAGE SECTION.
 01  THE-TABLE.
     05  THE-NUM             PIC 9(6) OCCURS 100 TIMES
         INDEXED BY THE-IND.
 01  THE-SUM                 PIC 9(8).
```

```
 PROCEDURE DIVISION USING THE-TABLE THE-SUM.
 START-OUT.
     MOVE 0 TO THE-COUNT
     PERFORM ADD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     MOVE THE-COUNT TO THE-SUM
     GOBACK

 ADD-EM.
     ADD THE-NUM(THE-IND) TO THE-COUNT

* ************************************************************
* ************************************************************


C ************************************************************
C THIS IS THE FORTRAN SUBPROGRAM MENTIONED IN THE
C  PRECEEDING COBOL MAIN PROGRAM
C ************************************************************
C
$CONTROL LIST,MAP,LOCATION,STAT
       SUBROUTINE FORADD(INMAT,SUM)
       CHARACTER*6 INMAT(100)
       CHARACTER*8 SUM
       INTEGER*4 OUTPUT
       OUTPUT = 0
       DO 60 I=1,100
  60   OUTPUT = OUTPUT + JNUM(INMAT(I))
       SUM = STR(OUTPUT,8)
       RETURN
       END

* ************************************************************
* ************************************************************


* ************************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES BINARY DOUBLE WORDS TO A COBOL SUBPROGRAM
*  AND A FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
```



```
* THE SUBPROGRAMS EACH ADD 100 BINARY DOUBLE WORDS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK   9942 CPU MILLISECONDS
*    FORTRAN TOOK 2499 CPU MILLISECONDS
* ************************************************************
$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS TESTFORD
 PROGRAM-ID. FORCOBDB.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                  PIC S9(9) COMP.
 01  STAMPE                  PIC S9(9) COMP.
 01  STAMPD                  PIC 9(6).
 01  THE-SUM                 PIC S9(9) COMP.
 01  THE-SUMD                PIC 9(8).
 01  THE-COUNT               PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM             PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 DOUBLE ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
```

```
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM FOR-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
     STOP RUN
     .
 LOAD-EM.
     ADD 5 TO THE-COUNT
     MOVE THE-COUNT TO THE-NUM(THE-IND)
     .
 COB-SHOT.
     CALL "COBADD" USING THE-TABLE THE-SUM
     .
 FOR-SHOT.
     CALL "FORADD" USING THE-TABLE THE-SUM
     .
* ************************************************************
* ************************************************************


* ************************************************************
* THIS IS THE COBOL SUBPROGRAM MENTIONED IN THE
*  PRECEEDING COBOL MAIN PROGRAM
* ************************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBSB
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT                PIC S9(9) COMP.
 LINKAGE SECTION.
 01  THE-TABLE.
     05  THE-NUM              PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 01  THE-SUM                  PIC S9(9) COMP.
 PROCEDURE DIVISION USING THE-TABLE THE-SUM.
 START-OUT.
     MOVE 0 TO THE-COUNT
```



```
     PERFORM ADD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     MOVE THE-COUNT TO THE-SUM
     GOBACK

 ADD-EM.
     ADD THE-NUM(THE-IND) TO THE-COUNT

* ************************************************************
* ************************************************************


C ************************************************************
C THIS IS THE FORTRAN SUBPROGRAM MENTIONED
C  IN THE PRECEEDING COBOL MAIN PROGRAM
C ************************************************************
C
$CONTROL LIST,MAP,LOCATION,STAT
       SUBROUTINE FORADD(INMAT,SUM)
       INTEGER*4 INMAT(100)
       INTEGER*4 SUM
       INTEGER*4 OUTPUT
       OUTPUT = 0
       DO 60 I=1,100
  60   OUTPUT = OUTPUT + (INMAT(I))
       SUM = (OUTPUT)
       RETURN
       END

* ************************************************************
* ************************************************************


* ******************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES BINARY DOUBLE WORDS TO A COBOL SUBPROGRAM
*  AND A FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
* THE SUBPROGRAMS EACH ADD 100 BINARY DOUBLE WORDS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
```

```
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* THE FORTRAN SUBPROGRAM WAS THE SAME AS THE ONE USED
*  IN THE PRECEEDING TEST
* I CHANGED THE COBOL SUBPROGRAM TO SHOW HOW COBOL
*  COULD BE SPEEDED UP USING DIFFERENT CODING
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK   6527 CPU MILLISECONDS
*    FORTRAN TOOK 2482 CPU MILLISECONDS
* ********************************************************
$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS TESTFORD
 PROGRAM-ID. FORCOBDB.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                  PIC S9(9) COMP.
 01  STAMPE                  PIC S9(9) COMP.
 01  STAMPD                  PIC 9(6).
 01  THE-SUM                 PIC S9(9) COMP.
 01  THE-SUMD                PIC 9(8).
 01  THE-COUNT               PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM             PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 DOUBLE ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
```



```
     DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM FOR-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
     STOP RUN
     .
 LOAD-EM.
     ADD 5 TO THE-COUNT
     MOVE THE-COUNT TO THE-NUM(THE-IND)
     .
 COB-SHOT.
     CALL "COBADD" USING THE-TABLE THE-SUM
     .
 FOR-SHOT.
     CALL "FORADD" USING THE-TABLE THE-SUM
     .
* ********************************************************
* ********************************************************


* ********************************************************
* THIS IS THE IMPROVED COBOL SUBROUTINE MENTIONED IN THE
*  PREVIOUS COBOL MAIN PROGRAM.  IT SHOWS THAT A PROGRAM
*  CODED LOOP CONTROL CAN BE MORE EFFICIENT THAN A
*  COMPILER CONTROLLED LOOP IN SOME CASES
* UNLESS HARD PRESSED FOR PERFORMANCE, I WOULD NOT
*  USUALLY PREFER MY OWN LOOP CONTROL
* ********************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBXB
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT               PIC S9(9) COMP.
 01  THE-IND                 PIC S9999 COMP.
 LINKAGE SECTION.
 01  THE-TABLE.
```

```
    05  THE-NUM              PIC S9(9) COMP OCCURS 100 TIMES.
01  THE-SUM                 PIC S9(9) COMP.
PROCEDURE DIVISION USING THE-TABLE THE-SUM.
START-OUT.
    MOVE 0 TO THE-COUNT
    MOVE 1 TO THE-IND
    .
ADD-EM.
    ADD THE-NUM(THE-IND) TO THE-COUNT
    ADD 1 TO THE-IND
    IF THE-IND < 101
        GO TO ADD-EM
    .
    MOVE THE-COUNT TO THE-SUM
    GOBACK
    .
* ************************************************************
* ************************************************************


<< THIS SUBPROGRAM ALLOWS BINARY SEARCH OF PB-RELATIVE >>
<<  CODE. IT REQUIRES A FIXED LENGTH ARGUMENT AND      >>
<<  SENDS BACK A FIXED LENGTH RESULT.                  >>

$CONTROL SUBPROGRAM,SEGMENT=FSEARCH
<< KEPT AS SEARCHFS >>
<< FIXED LEN ARG (WORDS) AND FIXED LEN RESULT (WORDS) >>
BEGIN PROCEDURE FINDFIXED(FOUND,ARG,RESULT);
INTEGER FOUND;
INTEGER ARRAY ARG,RESULT;
BEGIN
EQUATE ARG'WLEN = 2;
EQUATE RESULT'WLEN = 17;
EQUATE STEP'SIZE = ARG'WLEN + RESULT'WLEN;
EQUATE ARG'BLEN = ARG'WLEN * 2;
EQUATE RESULT'BLEN = RESULT'WLEN * 2;
EQUATE NUM'COMPARES = 5;
<< FOR NUM'COMPARES:                                  >>
<< IF TOO LARGE, WASTED TIME; IF TOO SMALL, NO-HITS   >>
<< GENERAL GUIDELINE:                                 >>
<< IF NUM'COMPARES ** 2 IS LESS THAN THE NUMBER       >>
<<  OF ARGUMENTS TO BE SEARCHED, THE SEARCH WILL      >>
```



```
<<  NOT SUCCEED IN ALL CASES                          >>
<< IF (NUM"COMPARES -1) ** 2 IS GREATER THAN THE      >>
<<  NUMBER OF ARGUMENTS TO BE SEARCHED, THERE WILL    >>
<<  BE SOME WASTED COMPUTER CYCLES                    >>
INTEGER POS,DISP;
BYTE POINTER RESULTB,BARG;
TOS:=@ARG;
ASSEMBLE (LSL 1);
@BARG:=TOS;
FOUND:="NN";
TOS:=@RESULT;
ASSEMBLE (LSL 1);
@RESULTB:=TOS;
TOS:=1;
ASSEMBLE (LSL NUM'COMPARES);
DISP:=TOS;
POS:=@START'DATA + ((DISP - 1) * STEP'SIZE);
WHILE DISP <> 0 DO
  BEGIN
    DISP:=DISP/2;
    IF POS >= @END'DATA THEN
      POS:=POS - (DISP * STEP'SIZE)
      ELSE
      BEGIN
        TOS:=@BARG;
        TOS:=POS;
        ASSEMBLE (LSL 1);
        TOS:=ARG'BLEN;
        ASSEMBLE (CMPB PB);
        IF = THEN
          BEGIN
            DISP:=0;
            FOUND:="YY";
            TOS:=@RESULT;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN;
            ASSEMBLE (LSL 1);
            TOS:=RESULT'BLEN;
            ASSEMBLE (MVB PB);
          END
          ELSE
          IF < THEN POS:=POS-(DISP * STEP'SIZE)
```

```
          ELSE POS:=POS+(DISP * STEP'SIZE);
      END;
  END;
ASSEMBLE (EXIT 3);
START'DATA:
<< START CONS >>
ASSEMBLE (CON "AB  RESAB                          ");
ASSEMBLE (CON "BB  RESULT     BB                  ");
ASSEMBLE (CON "CC  RESULT    CC              !!   ");
ASSEMBLE (CON "CEF RES CEF                        ");
ASSEMBLE (CON "CEFARESULT CEFA                  **");
ASSEMBLE (CON "DE  RDE                            ");
ASSEMBLE (CON "DEF RDEF                           ");
ASSEMBLE (CON "GHIJRESULT FOR GHIJ----            ");
ASSEMBLE (CON "GHK RESULT FOR GHK.................");
ASSEMBLE (CON "KKL RESULT FOR KKL                 ");
END'DATA:
END;
END.

* ********************************************************
* ********************************************************


<< THIS SUBPROGRAM ALLOWS BINARY SEARCH OF PB-RELATIVE >>
<<  CODE. IT REQUIRES A FIXED LENGTH ARGUMENT AND      >>
<<  SENDS BACK A FIXED LENGTH RESULT.                  >>
<< THE RESULT IS STORED IN CODE AS A VARIABLE LENGTH   >>
<<  ENTITY AND THE SUBPROGRAM NEED NOT WASTE SPACE     >>
<<  WITH TRAILING BLANKS.  THIS CAN BE SIGNIFICANT.    >>

$CONTROL SUBPROGRAM,SEGMENT=VSEARCH
<< KEPT AS SEARCHVS >>
<< FIXED LEN ARG (WORDS) AND VARIABLE LEN RESULT (WORDS) >>
BEGIN PROCEDURE FINDVARIABLE(FOUND,ARG,RESULT);
INTEGER FOUND;
INTEGER ARRAY ARG,RESULT;
BEGIN
EQUATE ARG'WLEN = 2;
EQUATE RESULT'WLEN = 1;
EQUATE STEP'SIZE = ARG'WLEN + RESULT'WLEN;
EQUATE ARG'BLEN = ARG'WLEN * 2;
```



```
EQUATE RESULT'BLEN = RESULT'WLEN * 2;
EQUATE NUM'COMPARES = 5;
<< FOR NUM'COMPARES:                                   >>
<< IF TOO LARGE, WASTED TIME; IF TOO SMALL, NO-HITS    >>
<< GENERAL GUIDELINE:                                  >>
<< IF NUM'COMPARES ** 2 IS LESS THAN THE NUMBER        >>
<<  OF ARGUMENTS TO BE SEARCHED, THE SEARCH WILL       >>
<<  NOT SUCCEED IN ALL CASES                           >>
<< IF (NUM"COMPARES -1) ** 2 IS GREATER THAN THE       >>
<<  NUMBER OF ARGUMENTS TO BE SEARCHED, THERE WILL     >>
<<  BE SOME WASTED COMPUTER CYCLES                     >>
INTEGER DSTART,DEND,POS,DISP;
BYTE POINTER RESULTB,BARG;
TOS:=@ARG;
ASSEMBLE (LSL 1);
@BARG:=TOS;
FOUND:="NN";
TOS:=@RESULT;
ASSEMBLE (LSL 1);
@RESULTB:=TOS;
TOS:=1;
ASSEMBLE (LSL NUM'COMPARES);
DISP:=TOS;
POS:=@START'DATA + ((DISP - 1) * STEP'SIZE);
WHILE DISP <> 0 DO
  BEGIN
    DISP:=DISP/2;
    IF POS >= @END'DATA THEN
      POS:=POS - (DISP * STEP'SIZE)
      ELSE
      BEGIN
        TOS:=@BARG;
        TOS:=POS;
        ASSEMBLE (LSL 1);
        TOS:=ARG'BLEN;
        ASSEMBLE (CMPB PB);
        IF = THEN
          BEGIN
            DISP:=0;
            FOUND:="YY";
            TOS:=@RESULT;
            ASSEMBLE (LSL 1);
```

```
            TOS:=@DSTART;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN;
            ASSEMBLE (LSL 1);
            TOS:=2;
            ASSEMBLE (MVB PB);
            TOS:=@DEND;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN*2+1;
            ASSEMBLE (LSL 1);
            TOS:=2;
            ASSEMBLE(MVB PB);
            TOS:=@END'DATA+DSTART;
            ASSEMBLE (LSL 1);
            TOS:=2*(DEND-DSTART);
            ASSEMBLE (MVB PB);
          END
          ELSE
          IF < THEN POS:=POS-(DISP * STEP'SIZE)
            ELSE POS:=POS+(DISP * STEP'SIZE);
      END;
  END;
ASSEMBLE (EXIT 3);
START'DATA:
<< START CONS >>
ASSEMBLE (CON "AB  ",0000);     << NO PREVIOUS LEN  >>
ASSEMBLE (CON "BB  ",0003);     << PREVIOUS LEN 03W >>
ASSEMBLE (CON "CC  ",0010);     << PREVIOUS LEN 07W >>
ASSEMBLE (CON "CEF ",0025);     << PREVIOUS LEN 15W >>
ASSEMBLE (CON "CEFA",0029);     << PREVIOUS LEN 04W >>
ASSEMBLE (CON "DE  ",0046);     << PREVIOUS LEN 17W >>
ASSEMBLE (CON "DEF ",0048);     << PREVIOUS LEN 02W >>
ASSEMBLE (CON "GHIJ",0050);     << PREVIOUS LEN 02W >>
ASSEMBLE (CON "GHK ",0060);     << PREVIOUS LEN 10W >>
ASSEMBLE (CON "KKL ",0076);     << PREVIOUS LEN 16W >>
ASSEMBLE (CON -1,-1 ,0083);     << PREVIOUS LEN 07W >>
END'DATA:
ASSEMBLE (CON "RESAB "                          ); << 03W >>
ASSEMBLE (CON "RESULT     BB "                  ); << 07W >>
ASSEMBLE (CON "RESULT    CC              !! "   ); << 15W >>
ASSEMBLE (CON "RES CEF "                        ); << 04W >>
ASSEMBLE (CON "RESULT CEFA                   **"); << 17W >>
```



```
ASSEMBLE (CON "RDE "                            ); << 02W >>
ASSEMBLE (CON "RDEF"                            ); << 02W >>
ASSEMBLE (CON "RESULT FOR GHIJ---- "            ); << 10W >>
ASSEMBLE (CON "RESULT FOR GHK................." ); << 16W >>
ASSEMBLE (CON "RESULT FOR KKL"                  ); << 07W >>
END;
END.

* ************************************************************
* ************************************************************


<< THIS ROUTINE PERFORMS AN AUTOREAD AGAINST A V/3000 >>
<<  FORM. IT IS CALLED WHENEVER YOU WISH TO HAVE THE  >>
<<  SCREEN READ AFTER THE USER HAS HIT A SOFT-KEY AND >>
<<  YOU DO NOT WISH TO FORCE HIM/HER TO HIT "ENTER"   >>
<<  TO TRIGGER THE READ.                             >>
<< IT IS CALLED JUST AS IF IT WERE "VREADFIELDS"     >>
<<  EXCEPT THAT THE NAME WOULD BE CHANGED            >>
<<                                                   >>
<<     CALL "IMMVREADFIELDS" USING VCONT-AREA        >>
<<                                                   >>
<<                                                   >>
$CONTROL SUBPROGRAM
<< KEPT AS IMMREAD >>
BEGIN PROCEDURE IMMVREADFIELDS(CONT);
INTEGER ARRAY CONT;
BEGIN
PROCEDURE VREADFIELDS(C);
INTEGER ARRAY C;
OPTION EXTERNAL;
CONT(55).(13:2):=%1;
VREADFIELDS(CONT);
CONT(55).(13:2):=0;
END;
END.

* 06*************************************************03
* ************************82*************************
```


:EOJ

User Control of Terminal Type Characteristics

David B. Mears
Hewlett-Packard Co.

I. INTRODUCTION

When operating a serial I/O device connected to a terminal port, there is a set of characteristics, including such things as echo and parity, which are collectively known as the terminal type characteristics for the port to which the device is connected. These characteristics determine how the terminal driver will communicate with the terminal.

Past HP3000 terminal drivers supported a specific set of terminal types, each with a different combination of settings for the characteristics. The different characteristics for each terminal type were achieved by checking for specific terminal types at certain points throughout the terminal driver code. The only way to add new terminal types was to change the driver code in several places and recompile. Thus, it was not practical to create new terminal types for users who needed something different from those already supported.

With the advent of the Advanced Terminal Processor terminal driver, the terminal type characteristics were consolidated into a table with one entry for each terminal type supported. The terminal driver now needs only to keep a pointer to the entry for the current terminal type and the code can make its decisions based on the values in the terminal type entry for that port. However, these entries are initialized at system startup from data in the code. Thus, to add new terminal types, it is still necessary to modify and recompile the code.

An enhancement to the ATP terminal driver moves the terminal type characteristics to a separate table for each port and adds the ability to initialize this table from data in a disc file. Several additional terminal type characteristics are also added to give a greater flexibility to control the terminal port. If a terminal type does not exist to meet the needs of a particular connection, a new one may be easily created to solve the problem, provided the set of characteristics available can satisfy the needs.

The remainder of this paper will discuss the terminal type characteristics that are supported and how they affect the terminal driver's control of the terminal port connection.

II. FLOW CONTROL

Flow control is the means by which the flow of data between the computer and the terminal device may be controlled to avoid sending data faster than the receiver can handle it. Flow control may be used to control the data flow in either direction, and may be implemented by either device. The first three flow control mechanisms discussed here are used for controlling the flow of data from computer to terminal. The last two are used for controlling the flow of data from terminal to computer.

A. Enquiry/Acknowledge

The Enquiry/Acknowledge protocol is used by the computer to control the flow of data to the terminal device. The terminal driver divides the data to be sent to the terminal into blocks of a certain size. Before each block is sent to the terminal, the driver will send an Enquiry character. If the device is ready to accept the block of data, it will send an Acknowledge character back to the computer. Upon receiving the Acknowledge, the terminal driver will send the block of data to the terminal. If instead, the terminal is not yet ready for the data, it will wait before sending the Acknowledge, and the driver will wait for the Acknowledge character before sending the data. Figure 1 shows a diagram of how the Enquiry/Acknowledge flow control works.

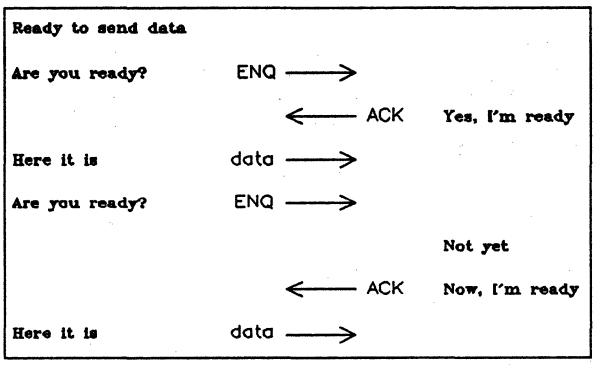


Figure 1
Enquiry/Acknowledge Protocol

Included in the Enquiry/Acknowledge flow control is a means for the driver to handle the circumstance in which the terminal device does not respond with an Acknowledge at all. When the driver sends the Enquiry character, it also starts a ten-second timer. Ten seconds is considered ample time

for the terminal to have processed the previous block of data and to be ready to accept the next.

If an Acknowledge is not received by the driver within the ten seconds, the driver will take one of three actions. If the assumption is made that the device actually sent the Acknowledge but that it was lost or garbled in the transmission, the terminal type characteristics can be set to have the driver send the block of data anyway. If a greater level of verification is desired, the characteristics may be set to have the driver send another Enquiry to the device and start another timer, repeating this action until an Acknowledge is received. The third option available is to report to the console that the Acknowledge was not received and wait for the condition to be resolved and for the device to send the Acknowledge before continuing to send data.

B. XON/XOFF

The XON/XOFF flow control mechanism is controlled by the terminal device. With this flow control mechanism, the terminal driver sends data to the device in a continuous stream as long as there is data to be sent. If the device encounters a situation in which it can no longer accept data, it notifies the computer by sending an XOFF character. Such situations include the device's internal buffer filling up, the device being put offline, or a printing device running out of paper. As soon as the driver receives and recognizes the XOFF character, it will suspend data transmission to the device. When the device is able to accept more data, it will send an XON character to the computer. When the XON is received and recognized, the driver will resume data transmission to the device. An illustration of XON/XOFF is shown in Figure 2.

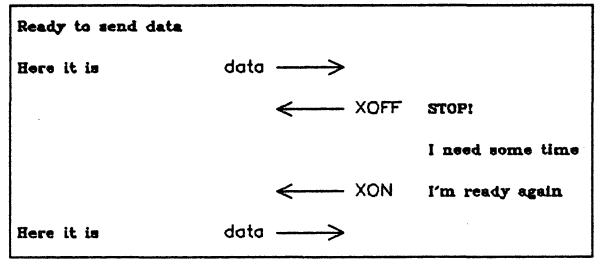


Figure 2
XON/XOFF Protocol

When devices are located remotely (such that there is normally no one attending the device), a time limit may be be set for the device to send an XON after it has sent an XOFF. Some conditions are considered temporary, such as the device's internal buffer filling up. After some period of time, the device will have been able to empty enough

characters from the buffer (printing them to the page, for example) that it can accept more data and thus will send an XON character. Other conditions are considered permanent, such as running out of paper, because the device cannot accept further data without intervention from an operator. In such a case, the time before the device will send the XON is indefinite without operator intervention. If an appropriate time limit is selected, the device will send the XON within the time limit for the temporary cases, but the time limit will expire for the permanent cases. When this happens, a message is sent to the console to indicate that the device needs attention.

C. Delay

The delay flow control mechanism is controlled by the driver. It is generally used with printing devices that do not have an internal buffer for data received. Such devices generally require extra time after carriage-motion-inducing characters, such as carriage return, line feed, and form feed, to physically move the carriage before further data may be accepted. With this flow control, the driver will wait after sending one of these three characters to the device for a specified amount of time before the transmission of data is resumed.

D. Read Trigger Character

There are two mechanisms that help control the flow of data from the terminal device to the computer. The first of these is the read trigger character.

The terminal driver can only process a single read or write request at a time. Sometimes, a program will want to request data from the device. To do this, it will issue a write request that will send a character string to the device to request the information desired. Then the program will issue a read request to read the information requested. If the device were to send the data immediately, it is likely the data would not be read completely because some of it would have been sent before the read request was issued. The read trigger character helps notify the device that the computer is ready to receive the data requested. When the device

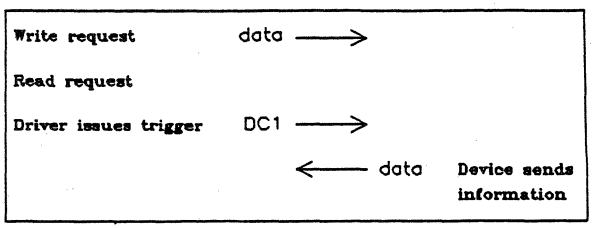


Figure 3
Read Trigger Handshake

gets the request for data, it waits until it has received the trigger character before sending the data. The read trigger character is sent by the driver after the read request has been received and the computer is ready to accept the data. Thus no data will be lost before the read has been started. Figure 3 shows the sequence of events.

E. Block Mode Reads

Under some circumstances, an application program will fill the terminal screen with a menu for data. Within this menu, there will be several fields for the user to enter data. Once all the data has been entered, the user will press the ENTER key to transmit the data to the computer. This type of transfer is known as Block Mode, where an entire block of data is transferred from the device to the computer, as compared to Character Mode, where each character is transmitted to the computer as it is typed by the user. An extra level of flow control is added to the read trigger to control a block mode transfer. A diagram is shown in Figure 4.

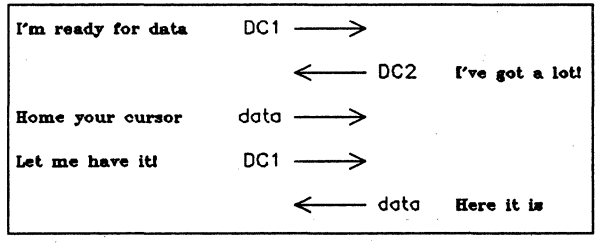


Figure 4
Block Mode Handshake

The read trigger character is sent to the device at the beginning of the read as usual. For a block mode transfer, the user will fill in the fields with the data and then press the ENTER key when ready to transmit the data on the screen. When the ENTER key is pressed, the terminal will send a block mode alert character to the computer. When the driver recognizes the alert character, it knows that the terminal has a large block of data to transmit. The driver can then prepare for the transfer, possibly by sending information to the device to move the cursor to the beginning of the data to transmit, or allocating a larger buffer area for receiving the data. When the driver is ready to receive the data, it will send a block trigger character to the device. When the device receives the block trigger character, it will begin transmitting the data on the display to the computer.

III. SPECIAL CHARACTERS

There are certain characters, usually within the unprintable control part of the ASCII character set, which are used not as data but to control the data passing between the terminal and the computer. Three of these characters are used to get attention for a console command, to specify that the last character input should be deleted, and to specify that the entire line entered so far should be deleted. There are also characters used to terminate the input of a line of data, and to send a subsystem break interrupt to an application program. Other characters may be specified to be ignored and not be placed within the data passed to the program.

IV. STRIPPING OF OTHER CHARACTERS FROM INPUT

At times, some of the special control characters do not have their normal control functions. Options are available to determine if those characters should then be stripped or included in the data passed to the program. If the XON/XOFF flow control is not enabled, then one of the terminal type options allows the user to specify that XON and XOFF be stripped from input. If a subsystem break character is entered when subsystem breaks are not enabled, the character may be removed from the input stream. If the console attention character is entered from a terminal which is not the console, it may be ignored or treated as a data character.

V. CONTROL

There are some general characteristics which affect the control of the terminal port connection. These are described in the following sections.

A. Echo

When the terminal is remote, in character mode, and local echo is disabled, characters typed on the keyboard are not placed in the display by the terminal. The computer must echo (or send back to the terminal) each character as it is received from the terminal for it to appear in the display. Even though each character must travel from the terminal to the computer and back to the terminal, the effect is that as characters are typed, they appear in the display instantly. When it is desired that the characters typed not be displayed, such as when entering a password or other sensitive information, echo may be disabled.

B. Form Feed During Output

Some devices do not recognize and act upon the form feed character in a useful manner. One of the terminal type characteristics available specifies that each form feed character in the outgoing data stream be replaced with a different character. Most often, this character will be the line feed character.

C. Line Feed During Input

Some devices do not provide an automatic wraparound when typing characters beyond the end of a display line. Without the wraparound, typing at the end of the line places each character upon all previously typed characters in the last position of the display line. To resolve this problem, the line feed character may be selected as a special character. If enabled, receipt of a line feed character will cause the driver to echo the line feed, write a carriage return character to the terminal, and remove the line feed from the input data. The result is to place the device's cursor or carriage at the beginning of the next display line. Thus, a line may be input that is longer than one display line, and will be displayed on more than one line.

D. Backspace Response

When a backspace character is entered, the previous character is deleted from the input stream. There are several options available to reflect this action on the device's display. In all cases, the backspace will be echoed to the device, provided echo has been enabled.

The most common response, used with CRT type displays, is simply to echo the backspace character. On the display, the cursor moves back one character position and is left at the character that was deleted. A second option is available for a few devices, such as the HP2600, which use the end-of-medium character to move the display cursor back. With this option selected, receipt of a backspace will cause the driver to send the end-of-medium character to the device to move the cursor back one character position.

Two options are available for printing type devices. For those devices which are able to move the print head backward, an option is available that will cause a line feed character to be sent to the device upon receipt of a backspace. This causes the print head to be positioned at the character deleted, but one line beneath it. As further data is input, it will line up properly, but will not be overprinted such that it would be unreadable. If several consecutive backspaces are entered, a line feed is generated

following only the first backspace. If more data is entered and then another backspace, another line feed will be generated. This keeps the data readable using the fewest possible line feeds. For devices which cannot physically move the print head backward, an option is a available in which the driver will respond to each backspace by sending the device a slash character followed by the character that was deleted. This gives the user the best means of knowing how many characters have been deleted from the input line.

A last option is available for use with CRT displays when it is desirable that characters deleted from the input be erased from the display. With this option, the driver will respond to backspace by sending the device a space and another backspace. Echoing the backspace received and sending the two characters generated results in erasing the deleted character from the display, leaving the cursor at that position. As part of this option, the driver will not echo any backspaces typed when the input buffer is empty, but will instead send a bell character to the device.

E. Parity

Parity is a means of verifying that data is transmitted between terminal and computer without error. The total number of information bits transmitted in a character is eight. These bits may all be used for the character code with two hundred fifty-six possible character codes available, or seven bits may be used for the character code (allowing one hundred twenty-eight characters) and the eighth bit used as a parity bit.

If seven data bits are used, there are four possible parity settings. The first is "force to zero." In this option, the parity bit is always made a zero. With the second option, "force to one," the parity bit is always made a one. The other two options are more useful for parity checking. These are odd and even parity checking. With these cases, the parity bit is set to either a zero or a one so that the total number of one bits out of the eight is either odd or even depending on the type of parity enabled. Figure 5 shows several examples of parity options.

	'$'	'*'
Force to zero	00100100	00101010
Force to one	10100100	10101010
Odd parity	10100100	00101010
Even parity	00100100	10101010

Figure 5
Parity Option Examples

The terminal type characteristics allow the choice of which parity option to use when the terminal port is allocated. This can occur under two circumstances: the port is opened by a program running from a different terminal port, or the port is speed-sensed in preparation to create a session on that port. When a port is speed-sensed, one of two parity options may be selected depending on whether the parity bit in the speed-sense character is a zero or a one. There are, therefore, three parity options (one for opening the port and two for speed-sensing the port) to select as part of the terminal type characteristics.

VI. PRINTER CONTROL

There are two characteristics specifically intended for use with printers connected to a terminal port. They are the specification of an initialization string for the printer, and the specification of "vertical format control" character sequences.

A. Initialization

When a printer exists in the spooled system printer category, different people may use the printer in different ways. In order to insure that one user who changes the printer characteristics, such as margins, tabs, or print density, doesn't affect the next user who doesn't want those choices of characteristics, an initialization string may be specified that will set up the printer to a known and repeatable state which is acceptable to most users. In such a state, users need not worry about resetting the device themselves. The initialization string is automatically sent to the printer by the terminal driver when the port is first opened. For spooled printers, this occurs at the beginning of every spool file sent to the device.

B. Vertical Format Control

Many line printers allow the use of vertical format control (VFC) to control the line spacing of output. Up to sixteen VFC channels may be supported by a device with each channel representing one or more places within the vertical page. As an example, one VFC channel might be set aside for top of page, another for bottom of page, another for every third of a page, etc. Some serial printers, such as the HP2631B, allow the use of vertical format control through a set of character sequences. To skip to a particular VFC channel, the corresponding character sequence is sent to the device. Among the terminal type characteristics, there is a flag to indicate if the device supports VFC. If it does,

the character sequences needed are included in the characteristics. When a program requests the device to skip to a specific channel, the driver will generate the character sequence needed by the device to skip to that channel.

VII. CONCLUSION

This paper has explained the meaning of "terminal type" as a set of characteristics which control the flow of data between computer and terminal. It has also explained what some of these characteristics are and how they affect the flow of data. The informed user should now be able to configure his devices to operate more efficiently with a greater degree of user friendliness. Also, future devices released by Hewlett-Packard should be easily configured to the system.

TECHNICAL ASPECTS OF THE PRIVATE VOLUME FACILITY

VICTOR MILONE

PRUDENTIAL REINSURANCE COMPANY

I. INTRODUCTION.

The Private Volume facility was implemented at Prudential Reinsurance Co. in the 1979-1980 time-frame. Since then, improvements in MPE have made the facility more consistent and easier to maintain. Interestingly, many of the techniques employed when the facility was first implemented are still useful today, in that the system manager can provide a more consistent and versatile facility to the users:

- More consistent, because minimal downtime will occur in the PV or system domain as a result of PV usage.
- More versatile, because different PV needs can be met across multiple computers with a minimum of complications.

II. THE PROCESSING ENVIRONMENT AT PRUDENTIAL REINSURANCE CO.

Various Reinsurance systems written in-house are processed locally in Newark, New Jersey. They were written in Cobol-3000 and some SPL-3000; they use IMAGE-3000 and KSAM-3000 and employ VIEW-3000 and TAPS (by Informatics Inc.) for terminal handling.

The Systems Division has been in a growth mode for the past five years, as follows:

1. Gradually upgraded from one HP-3000 Series II computer to two Series 64 computers. (DS-3000 will be used in the near future.)
2. Communication circuits are being leased from the Bell Co. to provide world-wide terminal access to the hosts in Newark.
3. On-line disc capacity has increased from roughly 150 megabytes to 3,120 megabytes using HP 7933 and 7925 disc drives.

Private volumes are now used extensively. Their usage has increased from one 7925 spindle and volume to ten spindles and roughly 22 volumes.

- In production, PV's handle large batch production and archived source code.
- In application development, PV's handle large test files that are

infrequently accessed.
- In operations and tech support, PV's handle the MPE account structure, the User Contributed Library, other special tools, and file transfer between computers.

III. GUIDLINES TO IMPROVE USE OF THE PV FACILITY.

Two areas in particular should be carefully managed in order to optimally utilize the PV facility. The primary area is correct maintenance of the system and PV domain directories. The second area is to define standards for mounting and dismounting PV's in a busy environment. If these areas are handled incorrectly, then (1) the support staff will spend unecessary time maintaining the facility in the face of outages and requests for PV's; (2) users will hesitate to use PV's due to loss of data integrity and inconveniences.

One unified set of streams should enable directories in both domains to be maintained without complications. The streams should exhibit the following characteristics:

1. The system manager can regenerate all directories for both domains on any computer by initiating a minimum number of streams.
2. All streams for both domains across all computers should reside in one group for quick modifications.
3. The Account Manager concept can be employed without sacrificing SM capability or disrupting the environment significantly.
4. Any account can be regenerated without regenerating other accounts unnecessarily.
5. PV's should be selectively executable on the appropriate computer(s).

A. Organization of the job streams (Figure A).

In an environment where directories need to be maintained across multiple computers which are in close proximity to one another, the streams should reside on a PV for transportability. This permits account structures which are identical across computers to be maintained in one file rather than many. In Figure A, the system domain is represented on the left, PV1 in the center, and PV2 on the right. The organization of five streams is shown for two types of accounts: Account #1 maintains some groups on the system domain and some on PV2; Account #2 maintains groups on PV2 only. All the streams reside in one group on PV1. The arrows in Figure A indicate which domain(s) the entries point to. The five streams perform the following functions:

1. The 'System Domain Control Stream' defines all accounts that

reside in the system domain of one computer. It then initiates streams at the Account Manager level to build each account's groups and users. If the entire system domain directory needs to be regenerated for one computer, the system manager would mount PV1 and stream the 'System Domain Control Stream.' This and the account-level streams it initiates would then regenerate the entire system domain directory for the one computer. Since this control stream does not handle PV's, they need not be mounted and dismounted during the regeneration. One of these control streams would exist for each computer.

2. The 'PV2 Control Stream' defines all accounts that are bound to PV2 only; this stream is executable on any computer. It too initiates streams at the Account Manager level to handle its groups. One of these PV control streams would exist for each PV regardless of the computer(s) it executes on. The system domain maintains entries for Account #2 which is bound to PV2.

3. The first 'Account #1 Stream' defines groups and users to the system domain only. Since the account is built by the 'System Domain Control Stream,' this organization effectively divides the System Manager and Account Manager functions, so the Account Manager need not possess SM capability to maintain his account. But if the system manager needs to regenerate the directories in both domains of any computer, he need not individually stream each account-level stream, since they are initiated from either a system domain control stream or from a PV control stream. Also, if the groups and users for one account are identical across multiple computers, only one account stream need exist, since it may be initiated from any number of control streams.

4. The second 'Account #1 Stream' defines all groups in Account #1 that are bound to PV2. Here too, the system domain maintains entries for the groups, all of which are bound to PV2. If other groups from the same account are to be bound to different PV's (say PV3), then another Account #1 Stream would be required for this, and so forth. If the Account Manager must change some group attributes, he can change the appropriate stream, mount the PV to be operated on, and then initiate the one stream which would regenerate all groups for that account on the one PV.

5. The 'Account #2 Stream' exemplifies a situation, where all groups for Account #2 exist on PV2. The stream, therefore, assigns the users to the system domain in addition to binding the appropriate groups to the PV.

B. Syntax of the job streams (Figures B & C).

Two conditions may exist when maintaining account structures: An entry is being created new, or an existing entry is being modified. In either case, the streams should satisfy both conditions without having to be rerun or modified. In the case of PV's and depending on the health of your system, entries may or may not exist in the PV or system domain. The streams should handle these possibilities as well.

The general flow of the 'PV2 Control Stream' (Figure B) is to define the volume set, define the accounts, and initiate the account-level streams. The following five commands define the sample stream in Figure B.

1. 'NEWVSET PV2...' is here in case an outage corrupts the system domain directory; it is also useful if volume class assignments are frequently changed. The PV should be logically dismounted for 'PURGEVSET...'; earlier versions of MPE had problems with a logically mounted volume. Another precaution is to turn VMOUNT OFF before purging the volume set.

2. 'ALTACCT SYS;VS=PV2.PUB.SYS:SPAN'. When an account already contains groups in the system domain, it is only necessary to SPAN the account to the PV.

3. 'NEWACCT...', 'ALTACCT...', 'ALTACCT...'. These three commands insure that the system and PV domain directories will be identical. If the account already exists, NEWACCT will be skipped; if not, the account will be built new. In any event, the two succeeding ALTACCT commands should never be skipped; they insure that both domains remain identical. Notice the absence of CONTINUE statements here. The directories should be identical in each domain, because from one version of MPE to another it is not clear which system domain attributes the SPAN parameter posts to the PV directory. Thus, 'ALTACCT X;VS=PV2.PUB.SYS:ALT' modifies the PV diriectory. Similarly, the 'ALTACCT X' without any VS parameter modifies the system domain directory.

In 1980, certain variants on this flow of commands would destroy parts of the system domain directory. A 'NEWACCT X;VS=PV2:SPAN' succeeded by two 'ALTACCT X;ACCESS=;VS=PV2:ALT' commands was especially devastating, especially with the ACCESS parameter preceding the VS parameter. Equally devastating was an 'ALTACCT X;VS=PV2' command without a SPAN or ALT parameter.

4. 'STREAM JPV2SYS' initiates the Account-level streams which define the groups and users in the SYS account to be bound to PV2. It could conceivably be delegated to an Account Manager. If this occured, the stream would probably reside in a different group for security reasons.

5. 'TELL ...' is very useful, because the output need not be proof-

read, if the message comes to the screen.

The syntax of the 'Account #2' stream (Figure C) is similar to the 'PV' control stream descibed above, except that it applies to the group-level entries instead of the account-level entries. The commands follow the same pattern of SPANS and ALTS as the PV Control stream. Since there are no system domain groups for Account #2, all users are defined at the end of the stream and are assigned to the system domain. User entries do not exist in PV directories.

C. Standards to mount and dismount private volumes.

Frequent PV mounts and dismounts invite trouble due to excess contamination entering the disc compartment. But since a certain number are inevitable, certain guidelines can minimize problems. The main guideline is to enter the correct commands in the correct sequence. Operator UDC's can greatly expedite the following sequence:

1. VSUSER
2. SHOWJOB can help determine whether any users are implicitly mounting or dismounting a PV.
3. DISMOUNT PV2 from any PV sessions.
4. LDISMOUNT PV2
5. DSTAT

Do not assume the logical dismounts were successful.

6. DOWN LDEV#
7. DSTAT
8. Spin down device.
9. Physically dismount volume.

The sequence to mount a PV is essentially the reverse of the above. If MPE does not recognize a volume, switch the PV 'off' and then 'on' to cause a recognizable IO interrupt. There are certain clarifications to be made in the above sequence of commands:

- It is unnecessary to turn VMOUNT OFF.
- There is no advantage in spinning down the drive before DOWNing the device (step 6).
- LDISMOUNT There is a possibility that the moment after DSTAT (step 5), a user may logically mount the PV.

IV. PRIVATE VOLUME DISC ERRORS.

PV disc errors can be caused by hardware, media, or power problems. They typically are not MPE problems. The diagnosis is difficult, because all three problems are manifested similarly and

can occur intermittently. Unfortunately, such problems typically result in long periods of down-time for diagnosis and resolution. Resolution often requires a full Reload, if the system domain was corrupted.

A. System domain disc errors.

Disc errors on portions of the system domain directory which are spanned to PV can cause a PV directory problem when attempting to access a PV file. In this case, try to purge the system domain entries. If the purge fails and other areas of the directory are corrupt, then a Reload is inevitable. The PV(s) will have to be regenerated after the Reload, although the PV data should be intact.

B. Types of private volume disc errors.

PV disc errors can occur in the PV directory, the Free Space Table, the user file area, or any other area of disc used by MPE. The servo code (on one particular face of a platter) may also have been corrupted due to age or contamination. An interactive LISTF,2 of the entire PV will indicate which file labels have been corrupted. Corrupted labels will show up scrambled on the screen, as each one is opened and read. A batch LISTF,2 will not necessarily indicate all corrupt file labels as an interactive one will. Similarly, any utilities such as Pvinit which print a formatted listing of the system tables will quickly indicate whether they have been corrupted. If any have, then the PV must be rebuilt with Pvinit, the accounts reset with the streams discussed above, and a PV Restore performed.

C. Causes of private volume disc errors.

To determine whether the disc errors were caused by hardware, media, or power problems, CE diagnostic techniques should be used. These include the stand-alone diagnostics, memory dumps, and log lists; the Systems Tables Reference Manual and CE diagnostic handbooks can be used to decode the relevant status codes. You will find a surprising similarity in the format of most device status codes, since they were obtained from the Device Information Table. Even the memory dump is not impossible to interpret, once you are familiar with the status codes. This type of diagnosis can save critical down-time when determining the type of problem.

Media problems can be minimized by keeping line printers away from PV spindles and from minimizing physical mounts to avoid contamination. Due to calibration tolerances which vary from disc spindle to spindle, a PV volume with bad media may work on one PV spindle but not on another.

Power problems can be minimized by insuring that your drives are powered through an appropriate isolation transformer.

V. PERFORMANCE OF PRIVATE VOLUMES (Figures D & E).

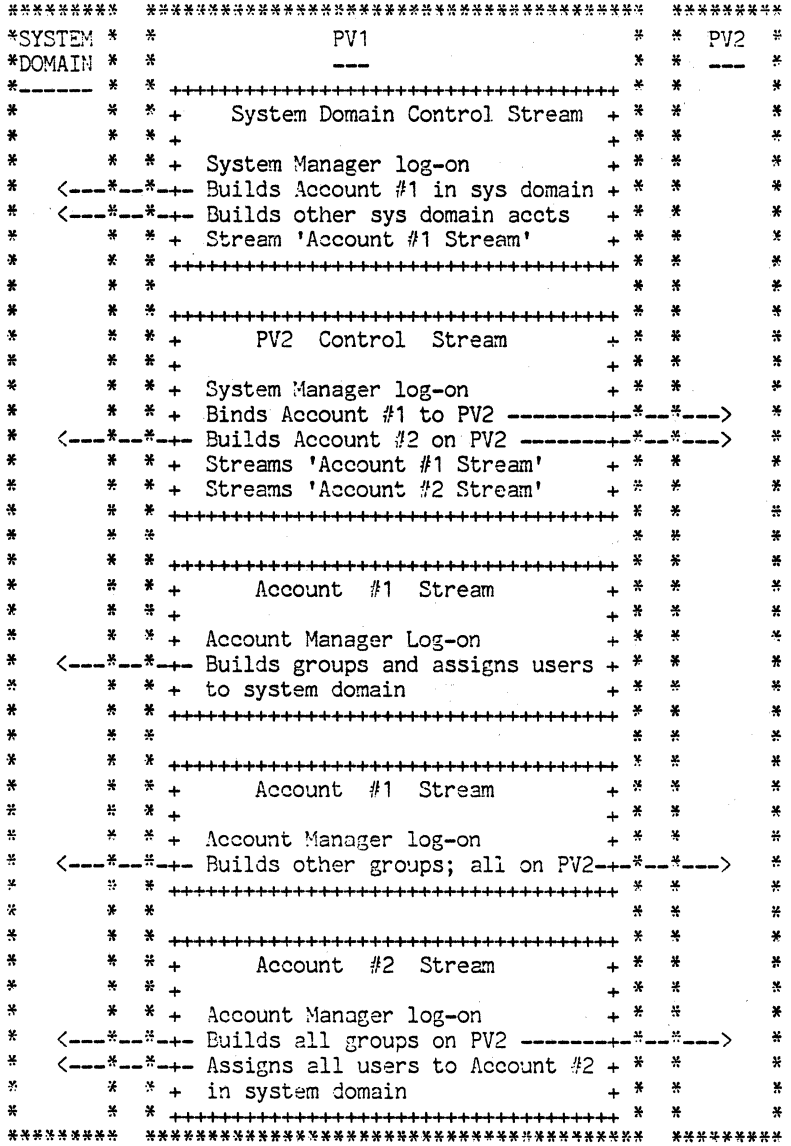
An important performance characteristic of PV's is that the time required to access PV files is similar to the time required to access system domain files. However, PV files take significantly longer to open than system domain files. The improvement pertains to the way in which MPE traverses from the system domain directory to the file index on the PV. When the file is first opened, the group directory entry in the system domain points to the group directory entry on the PV and then to the file index. After the first open, however, the group directory entry in the system domain points directly to the file index on the PV; the group entry on the PV is ignored. For infrequently opened files, this makes PV file access time similar to system domain file access time. This means that for files that are infrequently opened, performance in the PV domain should be similar to performance in the system domain.

The final two figures, from Hewlett Packard, indicate the similar access times for PV writes (Figure D) and system domain writes (Figure E) in tests under constant conditions. In migrating applications to PV's, we have not noticed appreciable degradations in access times or in total execute times.

VI. CONCLUSION.

The conclusion based on several years experience with the private volume facility is that for a growing computer division, PV's add significant flexibility and much needed disc capacity to a multi-processor environment. However, to fully utilize this potential, the facility should be carefully managed using guidlines such as the above. Further improvements should be obtainable through careful study and experimentation with the facility.

ORGANIZATION OF THE JOB STREAMS Figure A



PV2 CONTROL STREAM FIGURE B

```
!JOB JPV2CNTL,MANAGER.SYS,PV2GROUP
!DSTAT
!CONTINUE
!DISMOUNT PV2.PUB.SYS
!DSTAT
!CONTINUE
!PURGEVSET PV2
!CONTINUE
!NEWVSET PV2.    ; MEMBERS = PV2:HP7925
!DSTAT
!MOUNT PV2.PUB.SYS
!DSTAT
!CONTINUE
!ALTACCT SYS                ; VS = PV2.PUB.SYS:SPAN
!CONTINUE
!NEWACCT ACCTTWO,MGR        ; VS = PV2.PUB.SYS:SPAN
!ALTACCT ACCTTWO            ; VS = PV2.PUB.SYS:ALT  ; &
!                             PASS=                  ; &
! FILES = 150000             ; ACCESS =             ; &
! CAP   = AM,AL,GL,UV,ND,SF,IA,BA,PH,MR,DS,PM
!ALTACCT ACCTTWO            ; PASS=                 ; &
! FILES = 150000             ; ACCESS =             ; &
! CAP   = AM,AL,GL,UV,ND,SF,IA,BA,PH,DS,MR,PM
!COMMENT
!CONTINUE
!STREAM JPV2SYS.PV2GROUP
!CONTINUE
!STREAM JACCTTWO.PV2GROUP
!TELL MANAGER.SYS  JPV2CNTL IS COMPLETE  ! !
!EOJ
/
```

ACCOUNT #2 STREAM FIGURE C

```
!JOB JACCTTWO,MGR.ACCTTWO
!DSTAT
!MOUNT PV2.PUB.SYS
!DSTAT
!CONTINUE
!ALTGROUP PUB       ; VS = PV2.PUB.SYS:SPAN
!ALTGROUP PUB       ; VS = PV2.PUB.SYS:ALT; &
!                     PASS=                 ; &
! FILES =           ; ACCESS =              ; &
! CAP   = IA,BA,PH,DS
!ALTGROUP PUB       ; PASS=                 ; &
! FILES =           ; ACCESS =              ; &
! CAP   = IA,BA,PH,DS
!CONTINUE
!NEWGROUP DATA      ; VS = PV17.PUB.SYS:SPAN
!ALTGROUP DATA      ; VS = PV2.PUB.SYS:ALT; &
!                     PASS=                 ; &
! FILES =           ; ACCESS =              ; &
! CAP   = IA,BA
!ALTGROUP DATA      ; PASS=                 ; &
! FILES =           ; ACCESS =              ; &
! CAP   = IA,BA
!CONTINUE
!NEWGROUP JOB       ; VS = PV2.PUB.SYS:SPAN
!ALTGROUP JOB       ; VS = PV2.PUB.SYS:ALT; &
!                     PASS=                 ; &
! FILES =           ; ACCESS =              ; &
! CAP   = IA,BA
!ALTGROUP JOB       ; PASS=                 ; &
! FILES =           ; ACCESS =              ; &
!ALTUSER MGR        ; PASS=       ;HOME=PUB   ; &
! CAP   = AM,AL,GL,UV,IA,BA,ND,SF
!CONTINUE
!NEWUSER DB
!ALTUSER DB         ; PASS=       ;HOME=DATA  ; &
! CAP   = IA,BA,ND,SF,LG,UV
!TELL MANAGER.SYS  JACCTTWO IS  COMPLETE  ! !
!EOJ
/
```

PRIVATE VOLUMES
5000 Writes

Figure D

(Hewlett Packard)

Buffers:	2	4	6	8
Test #:				
1	23917	36693	36692	36691
2	35796	35795	35795	35794
3	35797	35796	35795	35795
4	35797	35795	35950	35794
5	35796	35796	35795	35794
6	35797	35795	35795	35794
7	35796	35796	35795	35794
8	35797	35796	35794	35795
9	35797	35795	35795	35794
10	35796	35796	35795	35794
Average Time	34608	35885	35900	35883

SYSTEM DOMAIN
5000 Writes

Figure E

(Hewlett Packard)

Buffers:	2	4	6	8
Test #:				
1	23288	35645	35668	35667
2	35419	35685	35418	35505
3	35485	36018	35417	35416
4	35418	35417	38307	35417
5	35419	35641	35996	35417
6	35419	35640	35728	35416
7	35419	35684	35417	35416
8	38375	35418	35417	35416
9	36063	35418	35417	35639
10	35419	35529	35417	35417
Average Time	34572	35519	35820	35472

A Name Indexing Subsystem for User Applications

Vaughn E. Minton
Third Judicial District Court
Topeka, Kansas

Introduction

Robert Womack outlined the following attributes which a name search technique should employ:

1) the ability to search on a partial or incomplete name
2) the ability to operate upon or display to a terminal user a group of similar names
3) the ability to search by a phonetic key
4) suitability for implementation using Image/3000

This author will explain how the Third District Court adapted the name indexing subsystem to their particular needs and will also add the following attributes to the four mentioned above:

5) the ability to find the person's (or entity's) case relationship
6) the ability to have multiple addresses and determine which is the primary address
7) the ability to have multiple names (or AKA's) for a given name
8) the adaptability of the name indexing subsystem to VIEW/3000 (or V-Plus)

Every on-line system will at one point in time require the ability to have the user type in a given name and determine if the name exists in the name data base. The designer now must make the decision of "How do I search the name data base for this one given name and how do I allow for incorrect spelling of names?" The Third District Court faced the same decision during the design of its first major on-line system "TRAMS". Because of this system's dependency on the user to enter a given name and retrieve the traffic case associated with that name, the designer's, Robert Womack and Robert Garvey, incorporated the first four attributes into the development of "Name Family."

Traditional Name Family

"Name Family", a generalized name indexing subsystem, allows the user to type in a name using View/3000 and determine if the name exists or should be added to the name data base. The logic for this procedure is as follows:

```
enter name
edit name
if no name exists
. enter name
if more than one name matches search criteria
  display names
  if name selected
    update name
  else
. . enter name
```

Image/3000

At first glance one will notice that the logic makes many assumptions. The first being that the system will incorporate View/3000 and Image/3000. By using Image/3000 the system must use some sort of a phonetic encryption algorithm to search for the names with a minimum of overhead on the 3000 and yet still allow for operator spelling errors. This "soundex" algorithm simply stated will take a name and return a soundex key; normally 6 bytes long. This key becomes the primary path in a detail set to allow the program to select the names that closely match the given name. This name selection process can also be more selective using a comparison technique to bring up the names that more closely match the given name. Mr. Robert Womack explains this process in greater detail in his paper presented at the Orlando, Florida conference. Other delimiters may be added to the matching criterion in order to more closely identify a range of names. As an example, the date of birth may be used to separate two names which are identical except for the personal descriptors. The following figure outlines the relationship between the soundex master and the name detail set.

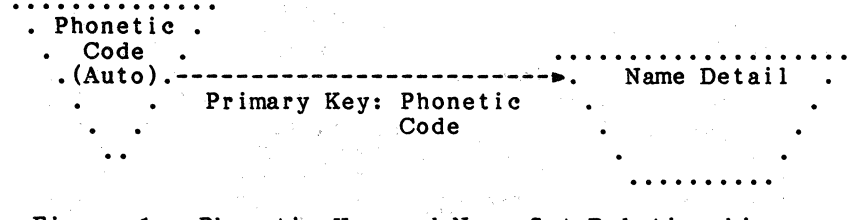


Figure 1. Phonetic Key and Name Set Relationship.

MINTON,VAUGHN,EUGENE,	more specific
MINTON,VAUGHN,E,	
MINTON,VAUGHN,,	
MINTON,V,,	
MINTON,,,	more general

Entity Control Block

With the birth of more complex systems at the Third District Court (and yes, more sophisticated users), the need to track roles that people have in a case and their relationship to that case became apparent. This role, the part played by a person in a particular case, may or may not be unique in all systems. For example, in the Court's Intake System roles are defined as "mother, father, brother, sister, client, etc." while in the Court's Clerk System the roles are far more complex as "first defendant, second defendant, attorney for first defendant, etc.". Please note that one and only one role may be defined for a particular case, but a person may have multiple roles in that case. The following example helps explain role relationship.

MINTON,VAUGHN,E,	attorney for 1st defendant
MINTON,VAUGHN,E,	attorney for 2nd defendant

Once the roles for a particular system have been defined, the designer must determine how to connect this role to a particular name for a particular case. The Third District Court utilizes a detail set entitled "ECB" (Entity Control Block) which contains the person's unique system sequential code, the case number and the person's role for that case. With the use of the ECB set the designer can now rapidly determine the person's role in a case and use that relationship as criterion for name selection. The reader should make careful note that the ECB set will enable the designer to list all people associated with a particular case and their role relationship, and on the other side of the coin, list all the cases associated with a particular name and the role relationship for each case.

The Third District Court enhanced the concept of roles by not using the alphabetical name of the role, but by assigning that role a unique alphanumeric code based upon the name of the role. The following role breakdown demonstrates the role name relationship.

The designer must also determine what information should be incorporated into the name data sets. The information stored in the name sets should be enough to fully describe the name, but also not too much to degrade system performance. Try to determine which information will be consistently updated and determine which information can be displayed with the name with minimal I/O thru Image. The following list shows an example of the data stored in the name set and address set. The name set would not include any address information because this information need only be required at the time of update. The name will be required multiple times through out the name selection process. Also in many external routines, the name may be required while the address information not be.

1. Name set -- soundex key
   system identification number
   name (50 bytes long)

2. Address set -- system identification number
   street address
   city
   state
   zip
   telepone number

View/3000

In the traditional approach a screen can be designed for the entering of the search name and other appropriate personal descriptors. If several names match the search criteria, then a second screen replaces the first which will list up to ten names at one time for the user to make a selection. If more than ten names, then the user may view additional names by hitting the proper function key signaling the program to either list the previous ten names or the next ten names. Upon the selection of a name, a third screen replaces the second which displays the name selected with all the related personal descriptors. At this point the user has the option to update the information or simply confirm the existing information as being correct.

The name entered on the screen must be edited with the format "last name,first name,middle name,suffix," with no exceptions allowed. Only the last name must be filled in in its entirety as the first, middle and suffix may be dropped for a more general name search. The external procedures for the "soundex" and "comparename" routines require this format. Do not despair, users will easily adapt to this method of entering names. The following example shows the generality of the name search technique.

```
0010100D     --   first party defendant
001          --   number of parties
   01        --   role code for defendant
     00      --   subcount
       D     --   abbreviation for defendant
```

In addition, this alphanumeric role serves the purpose of a sort item in that fewer hits to the ECB set were required to get the more frequently used roles. In this manner if all the first parties for a case were required to be listed, the program would select the entries in the ECB until the party count for defendants changed. The following list of roles will help to clarify this procedure.

```
0010100D -- first defendant
0010101D -- first defendant subcount one
0010300P -- first plaintiff
0020100D -- second defendant
```

In our example, only three hits to ECB set would be required to retrieve all the first parties in the case. This author cautions the reader that although this technique allows for fast retrieval of information, an expensive overhead cost is involved.

The ECB set becomes then the "Heart" of the system. Not only is the ECB set important to the name indexing subsystem, but other programs make full use of the set's capabilities. The Court's document print process relies completely on the roles stored in the ECB set. Separate external subprograms control the entering and updating of the roles in the ECB set. Prior to the entry of a role, the ECB is searched to determine if that role already exists, in which case the user is prompted to change the role or quit. As stated previously, expensive overhead but necessary.

Alias Names

Not every person identifies himself/herself by their proper name(ie, Bob, Robert; Jim, James; Sandy, Sandra; Betty, Elizabeth). As a result, the name given may actually be stored in the name data base without the user's knowledge. Thus, the user should be able to search on more than just one name at the same time. In other words, more than one name may be connected together and point to the same person resulting in alias names. In this manner, the user could enter "JONES,JIM,," as the search name and be able to locate the proper name which could be "JONES,JAMES,,".

As an example, in the Court's Intake System, the clientele includes juvenile offenders which result in more colorful names such as "SUPER FLY" and "SHADY LADY". With alias name capabilities the user could enter this nickname and locate the person's proper name. The user would also have the ability to inquire, add, update or delete alias names.

Under normal usage the user will only need to locate the "master" name for an individual. Because of this requirement, a sort field must be added to the name set. By using the codes "HS" for the master name, "IS" for alias names and blanks for no alias names, the designer will be assured of always retrieving the proper name with only one hit to the name set. Each name entered then under the alias name program will have its own soundex and will share the same identifying number.

Address Control Block

Even as a person may have more than one name, that same person or any person may have multiple addresses. Utilizing this concept the address set must be changed from a manual master to that of a detail set. By incorporating this technique into the name family schema, the changes are minimal. The operator would have the option to add or update the addresses in the address set.

However, this is not the normal procedure. In a court case at the Third District Court, the client may have several addresses with only one being the primary address. A defendant may give both his home address and work address for a particular case. In order that documents be sent to the correct address, the system must properly track the address pointers. This procedure can be accomplished by utilizing an address control block (ACB) to store the pointer to the address entry and the proper case numbers. Each address entered will have a pointer to this ACB set; the ACB set will then point to the case master.

During the document request process, the user looks at all the addresses for the name, then selects the address for the document. Since the address pointer is passed back to the calling program, this calling program will be able to locate the proper address. If no pointer is passed then the system uses the "default" address. Since addressess are case specific, a person can have multiple addresses for multiple cases.

Because of this importance placed on the ability to not only add addresses, but more particularly to update an address, a separate program was written to determine if that address could be updated. At the court the operator did not have the authority to change the address for a person in a

case unless that change was authorized by a lawyer or judge. Even then the same person could be related to another case and might need to have the adddress in that case stay the same. In other words, if a person has more than one case connection, then the user must not change that person's address. Only a new address may be added along with a pointer in the ACB set connecting the name and address. In addition to the entering of the address, the user is prompted to determine of the address should become the primary address for that case. If not a primary address, then is the address permanent or temporary. A separate batch program goes through the ACB set on a periodic basis to delete the temporary addresses.

VIEW Interface

Up to this point, this paper presented a name indexing system that required three screens in VIEW. As the complexity of the systems increased, the decision was made to condense those three screens into one. As a result of this move, the user could type in a name and without having the screen flip be shown a list of names that come close to matching the search criteria. This conversion takes place by checking the next form name and current form name in the comarea passed between the programs. This simple move not only added convenience for the user, but at the same time reduced overhead performance on the 3000. Since the same screen is shared by several programs, many of the screen edits are done by the programs.

Each program utilizes the eight function keys and re-defines them for each particular aspect of the name entry process. This author highly recommends this process because it increases the capabilities of VIEW. This author also encourages the use of the auto-read capability in view in order to read the screen contents when a function key has been pressed.

Dynamic Subprograms

At the Third District Court the name indexing subsystem utilizes some fifteen programs both in the USL and SL. The programs compiled into the USL should be those that are unique to that particular system. Those in the account SL (i.e. soundex routine) can be shared by multiple systems. Currently, all the programs for the name indexing subsystem are written in SPL. This has allowed the author to fully utilize the system intrinsics as well as adapt more easily to external subprograms. Also the SPL code produced smaller compiled code segments which results in a smaller stack requirement by the 3000. This enhanced system performance by reducing the amount of swapping done on the code segment. Remember that on the 3000 the code segment may be shared by several users while the data may not be shared. The

advantage being that with multiple users MPE will not have to make the code segment an overlay candidate since on the average at least one user will be accessing that code segment. Because name family utilizes one main block of subprogrms, this author compiled this main block into one segment in the USL. This resulted in faster response time because intra-segment transfer requires almost four times as less time than does inter-segment transfer.

Because the name family subsystem consists of several small dynamic subprograms, entry into the name family process can be accomplished at more that one level. The "call" to name family then becomes either "loud" or "quiet".

Calling name family "loud" would require that the name screen be retreived from the forms file, initialized, and displayed for the user to enter the search name. Once the initial download of the screen has been accomplished, the other name family programs need not repeat the same process. In short, why retreive a screen when that very screen is already displayed for the user?

In another perspective name family has multiple levels of execution. This process works in conjunction with the "loud/quiet" philosophy. A system flag passed amoung all programs in the system tells name family where the entry point should be and the "loud/quiet" status. One application may pass name family the search name while a call from the Main Menu would require name family to present the screen for the entering of the search name. In some applications, during a "quiet" call to name family, the screen may never change since the name search techniques work in the background.

Conclusion

In conclusion this author would like to emphasize that each analyst should carefully define their own name indexing problems. At the Third District Court there are six major on-line systems utilizing name family, but only two of them have an identical name indexing subsystem. During the system desing phase, the name indexing requirements were outlined so the name family code could be converted to meet those requirements. In this way the ECB and ACB were developed. The concept of the ACB required the talents of four analysts and almost an entire week of brainstorming to weigh the advantages and disadvantages of each technique. Use only the programs and techniques required.

Currently, the "Modern Name Family" consists of one view screen, a separate name data base, and all support programs written in SPL. This version can be adapted to almost any on-line system with very little conversion code. As with any data base, the DBUNLOAD/DBLOAD utilities should be used periodically.

If the proper guidlines have been established and followed without exception, then name family will become a valuable asset to any on-line system.

References

1. "A Generalized Name Indexing Method for Image Databases"
Robert L. Womack, IV
Robert B. Garvey
Witan, Inc.
Orlando, Florida Conference Proceedings

2. "Name Search Techniques"
Robert L. Taft
New York State Identification and Intelligence System
New York, New York

3. Court Systems Group
Third Judicial District Court
Topeka, Kansas
Robert B. Garvey, Analyst
Sally Henry, Analyst
Mike Cain, Programmer/Analyst
LiChun Ci, Programmer

Using VPLUS as a Driver For Computer-Assisted Instruction

Ronald D. Moore
Hewlett-Packard, Rolling Meadows, IL

INTRODUCTION

Most of my adult life has been in the classroom or in an educational environment. As an educator I was continually searching for techniques to enhance the learning process in my classroom. As a student I was pursuing a master's degree in computer science, and was told that to complete my degree I had to write a thesis. I didn't want to write on a topic that had no meaning to society, no meaning to the general public, no meaning to anyone except the professors on staff who would file the thesis in a bookcase to collect dust. The teacher, the rebel, the inventor and the desire to contribute within me forced me to write on a topic that I had to persuade the faculty to accept. I wanted to write on a topic that would help my students, one that would contribute to society and one that would satisfy the computer science faculty. The topic was the marriage of computer technology and education, which produced Computer- Assisted Instruction (CAI).

WHAT EDUCATION MEANS TO ME

Margaret Mead had this to say about education: "If we can't teach every student...something we don't know in some form, we haven't a hope of educating the next generation, because what they are going to need is what we don't know", which means that the role of educators and the total mass of society is to educate the next generation. CAI is one of the tools that can help to achieve that goal.

WHY CAI?

I chose CAI because of the positive effect that a good CAI package can have on a student. A good CAI package will build a positive self-image through the practice of starting with simple concepts that anyone could answer, then gradually progressing to more dificult topics. Before I expound on the positive effects of CAI, let me give you the negative and slip in how VPLUS will nullify one of the negatives.

DISADVANTAGES OF CAI

Time is a big negative for CAI. It takes about eight hours of development time for a package that will take a student only ten to fifteen minutes to complete. The physical limitations of your computer, the number of terminals, the type of terminals (CRT/hardcopy), and the amount of available disk space can all figure in negatively when placing CAI on your system. Another major drawback is the programming. Most educators are not programmers and most programmers don't have the expertise to write the material that is essential in developing good CAI. Here is where I slip in VPLUS.

VPLUS AND CAI

Using VPLUS as your driver program for CAI eliminates the problem of needing a programmer to make the ideas and concepts of the educator a reality in the form of a usable CAI package. VPLUS employs a free format screen design that is very easy to use and only a few program verbs need to be learned. VPLUS will decrease the time needed by the programmer. In fact, VPLUS could eliminate the need of a programmer. What could have taken two or more people to develop can now be accomplished by one. The advantages of using VPLUS are time, not needing a programmer, and using the edit specifications to "MATCH" answers to questions on the view screen.

The MATCH statement builds in a feature of answer checking that would be very difficult to program using a high level language such as BASIC or FORTRAN. I will *illustrate* this later in my sample application.

DISADVANTAGES OF VPLUS FOR CAI

There is basically only one disadvantage of using VPLUS for CAI, and that is the need for a VPLUS-compatible terminal. Most educational institutions will use character mode terminals which will not run in block mode and therefore will not support VPLUS. Some block mode terminals (non-HP) will use different escape sequences to perform a given terminal function. This means that the user is forced to use a terminal comparable to an HP2622A, HP2382A or better.

WHEN TO USE CAI

First of all, not all concepts are suitable for CAI. Some concepts are better taught with the aid of a film, slides, an overhead, or maybe even a handout. The phrase "a picture is worth a thousand words" may hold true when deciding what media to use when structuring learning material. If the concept you're developing learning material on is the atom, you might consider using a media better

suited for displaying pictures, such as a slide presentation. CAI is very good for teaching skills or concepts through the use of fill-in-the- blank, matching or true/false questions. It is not well suited for essay-type questions.

GETTING STARTED

If CAI is the appropriate media, then how do you get started? A CAI package should contain only one learning objective. Don't try to cover a lot of material at once. Break concepts into small units. For example, if you're developing a CAI package on integers, then develop four different packages: one each for addition, subtraction, multiplication and division. Start by defining your learning objective. (This module will help you learn how to add integers.) Now that you know what you want the user to learn, list all the steps needed to accomplish the task. Don't assume anything. State any rules and explain all symbols or special notations. This phase of preparation is done on paper; we are not ready to use the computer. Work a couple of problems to use as examples. (Definition--when adding integers and the signs are the same, add the two numbers and keep the given sign. Example: (+3) + (+2) = +5 (-4) + (-3) = -7).

DESIGNING A FRAME

A frame refers to what is displayed on the screen of the CRT. The first frame should state the objective and give directions on how to proceed. VPLUS provides a free format screen design, which enables the use of the edit keys and the cursor control keys. A frame is limited to 24 lines per frame and 80 characters per line. The fewer the lines or the less congested the screen, the better the readability. If a student is assigned a 1000 page book to read in a short period of time, the student might just look at the book and become discouraged enough to refuse to read it. On the other hand, a short story of 15 pages may not appear so overwhelming, and the student might thus attempt to complete the reading. The same type of analogy is true with CAI frames. It is better to use 20 frames to convey a concept than to cover it all on to 3 frames. The student will be more apt to understand 20 frames because of the ease of readability. The point here is the less material per frame, the better the frame.

PULLING IT ALL TOGETHER

A poem by Edgar A. Guest states:

"I'd rather see a sermon
than hear one any day
I'd rather one walk with me
than to just show me the way

The eye is a better pupil
and more willing than the ear
Your advice may be misleading
but examples are always clear."

I feel that this is what a CAI package should accomplish (hold your hand and walk with you) and that is what I'm going to do now. I can talk about how to develop a CAI package, but you can learn more from viewing the contents of a VPLUS CAI package. The following is instructions on how to get into VPLUS and a forms file listing with comments:

GETTING INTO VPLUS

LOG ON THE HP/3000.
:HELLO username.accountname
:RUN FORMSPEC.PUB.SYS
Formspec will display a forms menu screen:
FORMS FILE MENU (this will be at the top of the screen)
FORMS FILE NAME [ ] (you are to enter a file name inside the square brackets. It must begin with a letter and can't exceed eight characters. I recommend using five letter names or less. I'll show you why later.) You must now press the ENTER key from here on out, because you are in BLOCK MODE and the ENTER key signals the CPU that you are now sending data. Data is not transfered to the CPU until you press ENTER. You can use the EDIT FUNCTION KEYS and the CURSOR CONTROL KEYS. After pressing ENTER, you're in the
MAIN MENU
[ ] ENTER SELECTION
You will type in the letter "A" for add.
You must press the ENTER key.
This will take you to the next menu.
FORM MENU
FORM NAME [ ] (Now I'll explain why you want to use only five letters for the for name. Each form name will be the name of one frame. You will need a name convention. The convention is to append the numbers 01,02,03 04, etc. to the form file name. If the forms file name is TRIAL then the form names would be TRIAL01, TRIAL02, TRIAL03, etc..
Press the TAB key to get to the field on the screen.)
REPEAT OPTION [N] (Leave this field as is, press the TAB key)
NEXT FORM [C]
(Leave this field as is, press the TAB key)
NAME [ ]
(Place the name of the next form here. Example: if this is form TRIAL01 then the next form name TRIAL02 would go into this bracket)

Now press the ENTER key. When you press the ENTER key this time the message DESIGN NEW SCREEN AND PRESS ENTER will appear. This message will flash on the top left portion of a blank screen and will only be displayed for about thirty seconds. Then the screen will erase the message and leave you with a blank screen. The following is a listing of screen number one in my forms file which is named TRAIL. The name of the first screen has the form name of TRAIL01. Remember that a form name is the name of a screen and in CAI a screen is refered to as a frame. Hence the term screen and frame are synonymous or will be used interchangeable.

```
Form: TRIAL01
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL02
   Reproduced from:
```

<<This section corresponses to the FORM MENU>>

```
**********  **********  **********  **********  **********  **********  **********  **********
THIS SCREEN INTRODUCES THE OBJECTIVE AND/OR DIRECTIONS ARE GIVEN
**********  **********  **********  **********  **********  **********  **********  **********

                    USING THE HELLO COMMAND


THIS CAI PACKAGE WILL HELP YOU LEARN HOW TO USE THE 'HELLO' COMMAND.


IF YOU NEED HELP ON ANY FRAME, TYPE HELP AND PRESS THE 'ENTER' KEY.


IF YOU WISH TO EXIT THE CAI PACKAGE AT ANY TIME PRESS THE 'f8' KEY.


PLEASE PRESS THE 'ENTER' KEY TO CONTINUE OR 'f8' TO STOP.
**********  **********  **********  **********  **********  **********  **********  **********
**********  **********  **********  **********  **********  **********  **********  **********

THERE ARE NO FIELDS IN THIS FORM.
```

<<This is where EDIT SPECIFICATIONS will be typed in. You need to press the "f3" key at the top of your keyboard to get here.>>

<<The TEXT below the first line of stars and above the bottom line of stars (**** ****) is TEXT that I entered on the blank screen.>>

<<After designing your screen you must press the ENTER key >>

THIS PAGE INTENTIONALLY BLANK

<<After designing frame one (which did not have a *question*), you are prompted to enter form information for the next form, frame two.

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                              PAGE 5

Form: TRIAL02
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL03
   Reproduced from:


********** ********** ********** ********** ********** ********** ********** **********
THIS SCREEN WILL INTRODUCE THE PURPOSE OF HELLO
********** ********** ********** ********** ********** ********** ********** **********

:HELLO --> IS HOW TO START A SESSION

IT HAS THE FOLLOWING PARAMETERS:

:HELLO <SESSIONNAME,> USERNAME</USERPASSWORD>.ACCOUNTNAME</ACCOUNTPASSWORD>&
       <GROUPNAME</GROUPPASSWORD>> (AND A LIST OF KEYWORD PARAMETERS
       WHICH WE WILL NOT DISCUSS AT THIS TIME)

 NOTE: '&' IS USED AS A CONTINUATION CHARACTER WHEN A COMMAND IS TOO LONG
       TO FIT ON ONE LINE.

       < > INDICATE THAT ITS' CONTENT IS OPTIONAL.

  PRESS THE 'ENTER' KEY TO CONTINUE....
********** ********** ********** ********** ********** ********** ********** **********
********** ********** ********** ********** ********** ********** ********** **********

THERE ARE NO FIELDS IN THIS FORM.
```

<<Once again we do not have any edit specifications to supply, therefore frame three would look like: >>

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                              PAGE 6

Form: TRIAL03
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL04
   Reproduced from:

********** ********** ********** ********** ********** ********** ********** **********

YOU MAY HAVE NOTICED THAT THERE WERE ONLY TWO REQUIRED PARAMETERS;
          USERNAME AND ACCOUNTNAME (SEPERATED BY A PERIOD).

THE LEAST A USER WOULD NEED TO LOG ON THE SYSTEM IS A VALID
USERNAME AND A VALID ACCOUNTNAME.

IF A VALID ACCOUNT WAS 'MKGT' AND A VALID USER FOR THAT ACCOUNT
WAS A USER NAMED 'TOM' WHO IS ASSIGNED A HOME GROUP.  TOM COULD
LOG ON USING THE FOLLOWING 'HELLO' COMMAND:

:HELLO TOM.MKGT

NOTE: THE ':' IS A SYSTEM PROMPT, TOM DID NOT HAVE TO TYPE THE ':'.

PRESS THE 'ENTER' KEY TO CONTINUE.....
********** ********** ********** ********** ********** ********** ********** **********

THERE ARE NO FIELDS IN THIS FORM.
```

<<TRIAL04 is the first screen that ask a question. The key strokes used to create the field for the answer to a question is as followed:
Press the 'ESC' key
press the '[' key
Press the 'A' key
Press the space bar 22 times or enough times to create an area large enough to hold the answer
Press the 'ESC' key >>

<<To get to the bottom of the edit screen where the EDIT SPECIFICATIONS are given, you must press the 'f3' key >>

<<NOTE: To move from one field to another, you you must press the TAB key. >>

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                              PAGE 7

Form: TRIAL04
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL05
   Reproduced from:

********** ********** ********** ********** ********** ********** ********** **********


GIVEN: ACCOUNTNAME --> PAYROLL
       USERNAME    --> CLERK

ISSUE THE APPROPIATE 'HELLO' COMMAND TO LOG ON THE SYSTEM.

ENTER YOUR ANSWER

:A____________________


REMEMBER, IF YOU NEED HELP YOU CAN TYPE 'HELP'.
********** ********** ********** ********** ********** ********** ********** **********


Field: A
   Num: 1    Len: 22    Name: A                Enh: NONE  FType: P  DType: CHAR
   Init Value:
*** PROCESSING SPECIFICATIONS ***
UPSHIFT
JUSTIFY LEFT
IF MATCH HELLOb*CLERK.B*PAYROLL THEN CHANGE NFORM TO "TRIAL05"
   SET COUNTER TO 0
ELSE
    IF EQ "HELP" THEN CHANGE NFORM TO "HTRIAL04"
       SET COUNTER TO 0
    ELSE
        IF COUNTER LT 3 THEN SET COUNTER TO COUNTER + 1
           NE $EMPTY "PLEASE ATTEMPT A RESPONSE, YOU MAY TYPE'HELP'"
           MATCH HELLOb*CLERK.b*PAYROLL "NO TRY AGAIN. YOU MAY TYPE 'HELP'"
       ELSE FAIL "THE CORRECT ANSWER: 'HELLO CLERK.PAYROLL'"
```

<<NOTE: For every frame that ask a question the first two lines in the EDIT SPECIFICATIONS area should be:
UPSHIFT
JUSTIFY LEFT >>

```
<<NOTE:  MATCH for the correct answer.
         This will be on line three in
         the EDIT SPECIFICATIONS area.
IF MATCH condition THEN CHANGE NFORM TO "nextformname"
         Read the section of the VPLUS manual
         to find out more on the match
         statement.                          >>
```

```
FORMSPEC  VERSION B.03.03                        SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                            PAGE 8

Form: TRIAL05
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL06
   Reproduced from:


********** ********** ********** ********** ********** ********** ********** **********
VERY GOOD. LET'S PROCEED.

THE PREVIOUS EXAMPLE ASSUMED THAT 'TOM' HAD A HOME GROUP.

IF THE USER DOES NOT HAVE A HOME GROUP OR WANTS TO LOG ON TO
ANOTHER GROUP WITHIN THE SAME ACCOUNT HE CAN SUPPLY THE GROUPNAME.

EXAMPLE:  BILL IS A USER IN THE ENGIN ACCOUNT.
          HE HAS A HOME GROUP --> GBILL, BUT
          HE WANTS TO LOG ON TO THE GROUP --> GEDDIE.

BILL NEEDS TO SUPPLY THE GROUP --> GEDDIE, WHEN HE LOGS ON.


ANSWER FOR THE ABOVE EXAMPLE:

:HELLO BILL.ENGIN,GEDDIE

PRESS THE 'ENTER' KEY TO CONTINUE....
********** ********** ********** ********** ********** ********** ********** **********


THERE ARE NO FIELDS IN THIS FORM.
```

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                              PAGE 9

Form: TRIAL06
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIAL07
   Reproduced from:


********* ********* ********* ********* ********* ********* ********* **********


GIVEN:  JIM IS A VALID USER OF THE SALES ACCOUNT.
        JIM HAS A HOME GROUP OF GJIM, BUT HE WANTS
        TO LOG ON TO THE GROUP GJOE.


WHAT 'HELLO' COMMAND WOULD JIM HAVE TO ENTER:


ENTER YOUR ANSWER:

:A____________________


REMEMBER YOU MAY ASK FOR HELP BY TYPING 'HELP'.
********* ********* ********* ********* ********* ********* ********* **********


Field: A
   Num: 1    Len: 22    Name: A                 Enh: NONE  FType: P  DType: CHAR
   Init Value:
*** PROCESSING SPECIFICATIONS ***
UPSHIFT
JUSTIFY LEFT
IF MATCH HELLOb*JIM.b*SALES!,b*GJOE THEN CHANGE NFORM TO "TRIAL07"
   SET COUNTER TO 0
ELSE
    IF EQ "HELP" THEN CHANGE NFORM TO "HTRIAL06"
       SET COUNTER TO 0
    ELSE
        IF COUNTER LT 3 THEN SET COUNTER TO COUNTER + 1
           NE $EMPTY "PLEASE ATTEMPT A RESPONSE.  YOU MAY TYPE'HELP'"
           MATCH HELLOB*JIM.B*SALES!,B*GJIM "NO.  TRY AGAIN.  YOU MAY TYPE 'HELP'"
        ELSE FAIL "THE CORRECT ANSWER IS 'HELLO JIM.SALES,GJOE'"
```

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                              PAGE 10

Form: TRIALO7
   Repeat Option: N

   Next Form Option: C
   Next Form: TRIALO1
   Reproduced from:


********* ********* ********* ********* ********* ********* ********* **********
THIS IS THE FINIAL SCREEN
********* ********* ********* ********* ********* ********* ********* *********
THIS WILL CONCLUDE THIS CAI PACKAGE.

THE INTENT WAS TO ILLUSTRATE HOW TO DEVELOPE A CAI PACKAGE USE VPLUS.

IT BRINGS OUT THE FOLLOWING POINTS:

  1. HOW TO GET INTO FORMSPEC AND WHAT MENUES ARE AVAILABLE.

  2. HOW TO DESIGN A SCREEN.

  3. HOW TO USE THE EDIT SPECIFICATIONS.
     A.  USING 'MATCH', 'EQ', 'SET', 'FAIL' AND 'NE'
     B.  USING 'UPSHIFT' AND 'JUSTIFY LEFT'
     C.  USING 'IF', 'THEN' AND 'ELSE'
     D.  USING '$EMPTY' AND '$RETURN'
     E.  CHANGE NFORM TO "formname"

TO RUN YOUR "CAI" PACKAGE COMPILE YOUR FORMS FILE, EXIT AND
              "RUN ENTRY.PUB.SYS"

YOU MAY PRESS THE 'ENTER' KEY TO GO BACK THROUGH THIS CAI PACKAGE OR
PRESS 'f8' KEY TO EXIT
********* ********* ********* ********* ********* ********* ********* **********


THERE ARE NO FIELDS IN THIS FORM.

*******************************************************************************
*                                                                             *
*                          FORMSPEC  Version B.03.03                          *
*                        SAT, FEB 12, 1983,  6:48 PM                          *
*                                                                             *
*                                TRIAL.PUB.RON                                *
*                                                                             *
*******************************************************************************
Forms File Status
   Modified: SAT, FEB 12, 1983,  6:45 PM
   Compiled: SAT, FEB 12, 1983,  6:45 PM
   Requires 865  + 60 = 925  words  (add 500 for KSAMless fast forms file, or
                                     add 1300 for KSAMless slow forms file)

Head Form: TRIALO1
Default Display Enhancement: NONE
Error Enhancement: IU
Window Display Line: 24
Window Enhancement: HI


THERE ARE 1 SAVE FIELDS IN THIS FORMS FILE:

   Save Field: COUNTER                Length: 3          Data Type: DIG
      Init Value:


There are 9   forms in this forms file:

   Form              Num Fields   Num Lines   Next Form
   HTRIALO4          0            21          $RETURN
   HTRIALO6          0            20          $RETURN
   TRIALO1           0            23          TRIALO2
   TRIALO2           0            24          TRIALO3
   TRIALO3           0            23          TRIALO4
   TRIALO4           1            21          TRIALO5
   TRIALO5           0            21          TRIALO6
   TRIALO6           1            21          TRIALO7
   TRIALO7           0            24          TRIALO1
```

```
FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                             PAGE 2

Form: HTRIAL04
   Repeat Option: N

   Next Form Option: C
   Next Form: $RETURN
   Reproduced from:


********** ********** ********** ********** ********** ********** ********** **********
THIS IS A HELP SCREEN
********** ********** ********** ********** ********** ********** ********** **********

SO, YOU WANT SOME HELP. OK.

RIGHT AFTER THE ':' TYPE THE COMMAND 'HELLO'.

LEAVE A SPACE AFTER 'HELLO'.

TYPE THE USERNAME --> CLERK.

AFTER THE USERNAME TYPE A PERIOD '.'.

FOLLOW THE '.' WITH THE ACCOUNTNAME --> 'PAYROLL'.

YOU WOULD NORMALLY PRESS THE 'RETURN' KEY, BUT
SINCE YOU'RE IN BLOCK MODE PRESS THE 'ENTER'.


PRESS THE 'ENTER' KEY TO GET BACK TO THE QUESTION. "GOOD LUCK".
********** ********** ********** ********** ********** ********** ********** **********


THERE ARE NO FIELDS IN THIS FORM.



FORMSPEC  VERSION B.03.03                          SAT, FEB 12, 1983,  6:48 PM
FORMS FILE: TRIAL.PUB.RON                                             PAGE 3

Form: HTRIAL06
   Repeat Option: N

   Next Form Option: C
   Next Form: $RETURN
   Reproduced from:


********** ********** ********** ********** ********** ********** ********** **********
THIS IS A HELP SCREEN
********** ********** ********** ********** ********** ********** ********** **********

NEED A LITTLE HELP, DO YOU?

NO PROBLEM.  YOU GOT IT. HERE.

FOLLOWING THE ':' TYPE 'HELLO' THEN A SPACE.

NOW ENTER THE USERNAME --> 'JIM' FOLLOWED BY A PERIOD '.'.

THEN TYPE THE ACCOUNTNAME --> 'SALES'.

SINCE YOU MUST SUPPLY THE GROUP FOLLOW THE WORD 'SALES'
WITH out A COMMA ',' AND THE GROUPNAME --> GJOE.


PRESS THE 'ENTER' KEY TO GET BACKTO THE QUESTION.
********** ********** ********** ********** ********** ********** ********** **********


THERE ARE NO FIELDS IN THIS FORM.
```

SUMMARY

The last few pages was a listing of a forms file, with comments disbursted through out. Let's try to capture what was going on and how did we get to a given menu. To start we had to get into FORMSPEC. How do you do that?
:RUN FORMSPEC.PUB.SYS
Enter a forms file name, using five characters or less. Press the ENTER key all the while that you're in FORMSPEC. FORMSPEC runs in BLOCK MODE. The first menu is the FORM FILE MENU. The second menu is the MAIN MENU (you can get to the MAIN MENU at almost any time by pressing the 'f7' key). When creating a new forms file enter 'A' in the field to start a new forms file. If the forms file already exist then you can press 'f2' to go to the form that you want to modify or you can keep pressing 'f2' to go to the end of the forms file and enter a new form. The third menu is the FORM MENU. There are two things on this menu that you should enter. One is the name for the form that you're getting ready to design and second is the name of the next form. For the other fields take the default by pressing the TAB key. After enter the name for the form, you must press the ENTER key. You will find yourself staring at a blank screen. At this point you can enter the information for your CAI frame, using the CURSOR CONTROL keys and the EDIT keys on the top of your keyboard. The first frame should state the objective and give directions as to what to do next. Press the ENTER key when you finish designing your frame. If your frame had a question on it, the next menu that you'll see will be the FIELD MENU. Make your field specifications look like the field specifications in the form listing above. To get to the EDIT SPECIFICATIONS area at the bottom of the FIELD MENU, you must press the 'f3' function key at the top of the key board. Look at the examples in the forms listing above. From the examples and from reading the VPLUS manual, you should have a good idea of how to develop a CAI package. You will need a COUNTER, so you must create a SAVE FIELD just like the one in the listing. The COUNTER is to allow the user to make a mistake and the user is given a second or third chance. In the listing I gave the user three tries and on the third try if the user misses the question he is given the answer. The user must enter the correct answer before he can proceed. The next menu will be a FORM MENU again. From this point you will cycle through the following menus: (a) FORMS MENU (b) BLANK SCREEN --> designing a frame (c) FIELD MENU. On completing the last frame, the NEXT FORM should be the name of the first form. This will allow the user to go back through the CAI package. The user can EXIT the CAI

package at any time be pressing the 'f8' key. You can incorporate HELP SCREENS in your CAI package as in the example listing. When you have *completed* all of your frames and you are prompted with the FORM MENU enter the name of the HELP SCREEN. I used the letter 'H' in front of the FORM NAME to identify a given HELP SCREEN. Where 'HTRIAL04' is the HELP SCREEN for the frame/form named 'TRIAL04'. This should complete all the frames needed to bring your CAI package to life. Press the 'f7' to return to the MAIN MENU. Enter 'S' in the field to go to the SAVE MENU. Enter the information about the COUNTER just like in the listing. Press the 'f7' again. Now you're ready to compile your forms file. Enter 'X' in the field and press the Enter key. This will compile you forms file and point out any errors. If it compiles correctly then you can EXIT FORMSPEC by press the 'f8' key.

TESTING AND/OR RUNNING YOUR CAI PACKAGE

To test or run your CAI package you must run the entry program by typing :RUN ENTRY.PUB.SYS then press the RETURN key. You will be prompted for the FORMS FILE NAME and the BATCH FILE NAME. For the FORMS FILE NAME enter what you named your forms file and for the BATCH FILE NAME you can enter *anything* so let's type BATCH1. Why BATCH1? Well, because when you finish running your CAI package you will want to PURGE the CAI batch file to free up disc space. You may in the future :RUN REFSPEC.PUB.SYS against the batch file, then have a program read the batch file. This could reveal many things: (a) what questions the students are having difficulties with (b) what sections of the CAI package need to be revised. For now we'll call our batch file BATCH1 and PURGE it when you EXIT the package by pressing the 'f8' key. You could create a batch job and stream the CAI package:

```
!JOB caiuser.caiaccount
!RUN ENTRY.PUB.SYS
 caiformfilename
 caibatchfilename
!PURGE caibatchfilename
!EOJ
```

(let's say you keep it as CAIJOB01)

```
:STREAM CAIJOB1
```

This would keep the user from having to enter the above information and form having dozens of batch file on the system. Remember when running ENTRY to press the ENTER key.

WHERE DO YOU GO FROM HERE

Where to from here? To the drawing board. Now you know enough to be dangerous. Remember the quote, "A little knowledge is a dangerous thing". That is what you have now a little knowledge on USING VPLUS AS A DRIVER FOR CAI. So, what is your next step. One is to read the VPLUS manual and working through USING V/3000 (an introduction to forms design

for non-programmers). Don't try to reinvent the wheel. Find out what is available. There are some good packages in production. Next try your hand at developing a package. The first one is always the hardest. You must pay your dues. You'll learn from your mistakes. And remember never to assume. Try to bridge all gaps and show all the steps involved from getting from step A to step B. Be willing to experiment, "If we can't teach every student...something we don't know in some form, we haven't a hope of educating the next generation, because what they are going to need is what we don't know". We are creatures of habit. We learn from doing. The more we perform a given task the better we get. Use your CAI packages to drill success into the users. Try to help the user develop a positive self-image of himself. Don't try to stump the user but still make the package challenging and rewarding upon completion. Your course has been laid and a challenge placed before you. Should you so choose to accept this challenge, the fruits of your rewards will be as sweet as honey. It is a beautiful feeling to see the joy in a user when he has master a concept from completing a CAI package that you have developed. Don't you want to spread a little joy? If so, then accept the challenge and develop a CAI package.

IMPLEMENTATION OF TREE DATA STRUCTURE IN MPE

REZA NAZARI
HEWLETT-PACKARD COMPANY (ESR)
ROCKVILLE MARYLAND

INTRODUCTION:

The purpose of this paper is to explore the concept of tree data structures and to discuss the implementation of the algorithms in the MPE directory. This will give the HP3000 users a better understanding of the features and limitations of the MPE directory.

Tree data structures are often used to facilitate the file directory organization as well as the data base relationships. HP3000 has applied the tree data structure to organize the system's accounting structure and file directory information. The key file of the KSAM file is another application of the tree data structure in HP3000. Because of the complexity of the structure of the KSAM file and the limitations of the length of this paper, I will cover only the implementation of the MPE directory. My intent is to give the most benefit to general users of HP3000 rather than discuss detailed aspects of software which may be useful only to a limited group of users.

This paper is organized into four sections:

Section I gives background information about tree type data structures and familiarizes you with some of the terminology used throughout the paper. Section II illustrates the overall structure of the system directory and discusses the logical relationship of the elements in the directory. The physical layout of the directory on the system disc will be covered in Section III. Finally, Section IV summarizes the topics covered in the paper and will point out some of the directory considerations.

SECTION I: TREE DEFINITIONS

This section briefly describes the tree data structure and some elementary terminology used in this type of data structure, specifically those related directly or partially to the MPE directory.

Tree, simply defined, is a set of elements that are organized and related to each other through branches and laid out in a hierarchical fashion (Figure I-1). Elements in the tree are called nodes. The highest level of this hierarchy at the top is a single node called the root. Every node within the tree is related to the root directly or indirectly through the higher level nodes. The node at the lowest level of a branch is called

leaf. Every node between the root and the leaves is considered as a parent of one level lower nodes and as a child of the immediate higher level node. Each child cannot have more than one parent node. The root of the tree is considered Level 1 and it's chil- dren Level 2. Moving down the tree, the lowest level is Level n in the tree. The distance between the root and the leaves is called the depth of the tree. The number of nodes or children in each node is called the degree of the node.

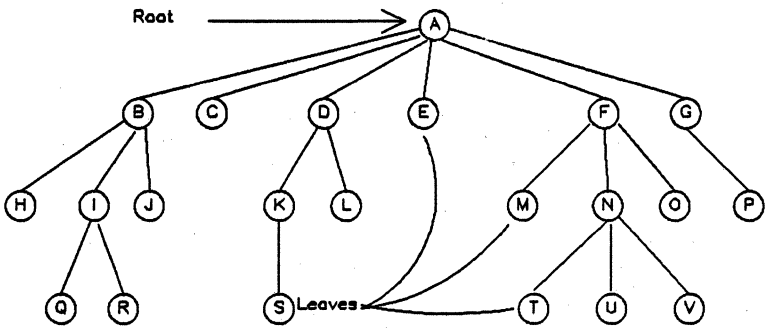


Figure I-1 Tree

A binary tree is described as a tree where every node including the root node can have at most two children. In other words, there is no node on the tree with a degree greater than two. The binary tree is an important type of tree structure which is more often used than other tree structures. Typical applications of binary trees can be found in file directories, symbol tables, and decision table algorithms. Figure I-2 gives an example of a binary tree with a depth of 4.

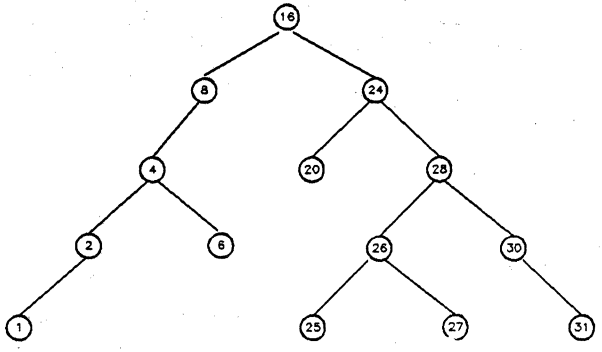


Figure I-2. Binary Tree

A balanced tree, which has more complex structure than a binary tree, is characterized by the fact that each node, including the root, should have the same number of children. A balanced tree is usually filled out starting from top to bottom and from left to right. A perfectly balanced tree is a tree where the distance of all leaves from the root are equal. Balanced trees are used mostly for file directory or data base management systems. The key file of KSAM/3000 is a good example of the balanced tree application. Figure I-3 shows a balanced tree with a degree of 4 and depth of 3.

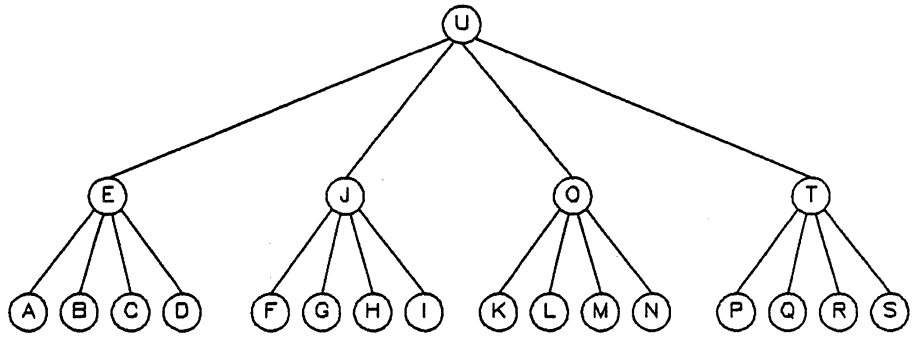


Figure I-3 Balanced Tree

B'tree is a balanced tree when the elements can only be placed in the leaf nodes.

SECTION II: DATA STRUCTURE OF THE DIRECTORY

This section describes the logical representation of the MPE directory, without considering at this time its physical layout on the system disc.

The MPE directory is a hierarchical data structure in a form of nested trees. The entire information about the accounts, users, groups, volume sets, volume classes, and file indexes of the system are stored in these trees. Figure II-1 shows an overview of the system directory. The highest level of this tree, which is considered the root of the tree, is the system account index block. The children of this node are account entry blocks. Technically, and for the purpose of simplicity, the system account index block and account entry blocks are referred to as a sub-tree. Up to five types of sub-trees can be found on this nested tree (tree is drawn upside down). Each account entry has pointers to the roots of two other sub-trees. These sub-trees contain the informa- tion about users and groups in each account. Every group in each ac- count again has pointers to the roots of two other sub-trees, i.e., one for file and the other for the volume set/volume class definitions in that group.

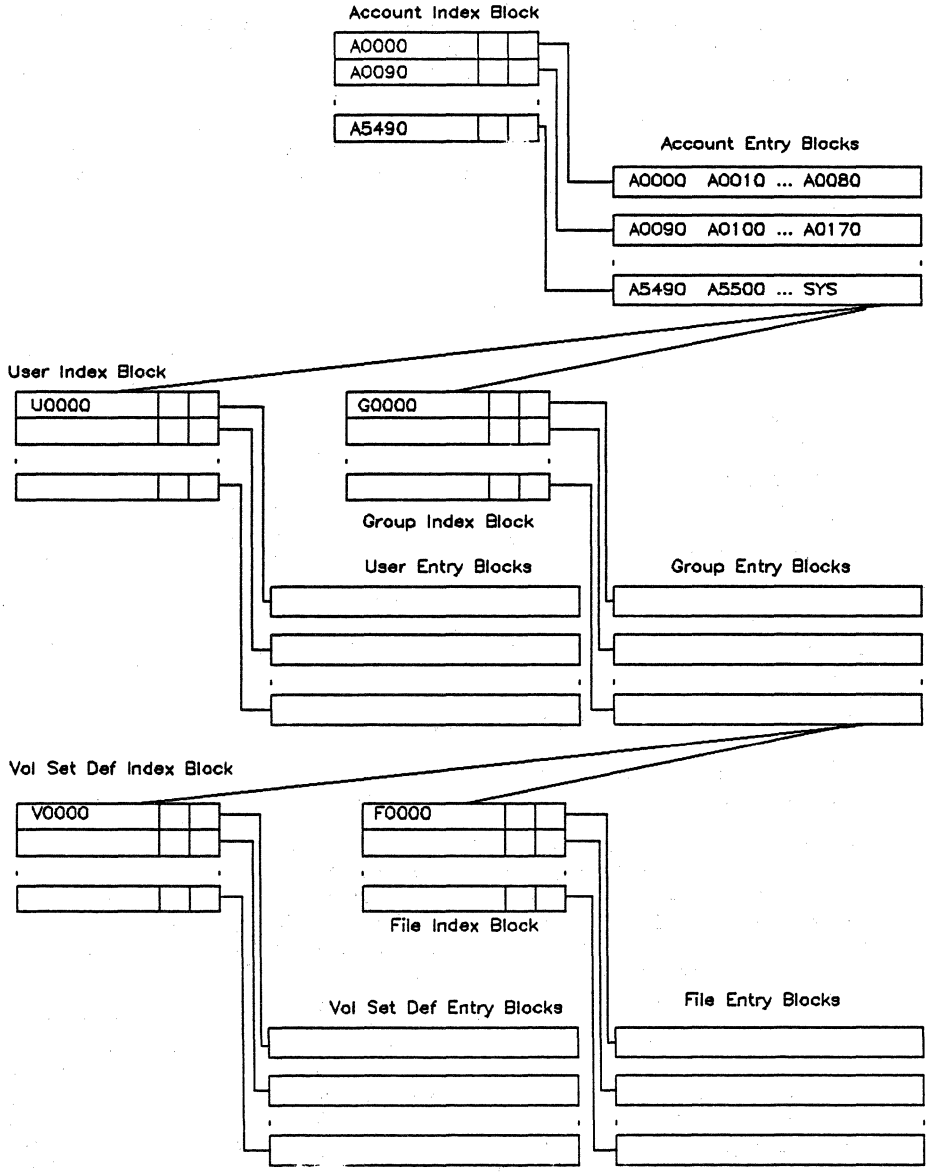


Figure II-1. Overview of Directory

The structure of these five sub-trees are identical. Each sub-tree has an index block and associated entry blocks. The index block does not contain any useful information about the system directory. The information about the directory is stored in the entry blocks of each sub-tree. Each sub-tree is considered as a balanced tree with depth of two. The degree of each tree depends on the size of the index block which will be discussed later. The index blocks and entry blocks within the sub-trees consist of an integral number of contiguous sectors (sector is 128 words). The index blocks and entry blocks of different sub-trees are not all the same size. The block sizes are established based on the size of the entries and occurrence of the entries.

Since the sub-trees are organized in the same fashion, we will describe the system account sub-tree in detail. Additional information about the other sub-trees will be provided if the reader desires to investigate further.

The system account index block is a unique block in the system and is created automatically to contain an index entry for SYS account which is entered in the account entry block as the first account in the system. The size of the account index block is three sectors. The first ten words of this block is called index block prefix. The prefix holds the information about the index block. Figure II-2 shows the format of the prefix and description of the fields. The rest of the index block contains up to 62 elements. The last two words of the index block are unused. Each element in the index block contains a key (four words) and a pointer to the entry block which holds the account entry for all accounts with a name equal or greater than the key value. The last word of each element in the index block is used as a counter and holds the number of the account entries in the account entry block. The account entry block contains the entire information about the ac- counts. The format of the account entry is in Appendix A-1. The size of the entry block is three sectors. Each account is thirty words and they are stored in alphabetical order. The maximum number of account entries that can be stored in each entry block is 12 and the last 4 words in the account entry block are wasted. Theoretically, you can store up to 744 accounts in the system.

The format and structure of other sub-trees in the directory are exactly the same, with the exception of the size of the index block and entry block. The following table shows the size and blocking factor of each block. The formats for the user, group, file, volume set definition, and volume class definition are in Appendix A.

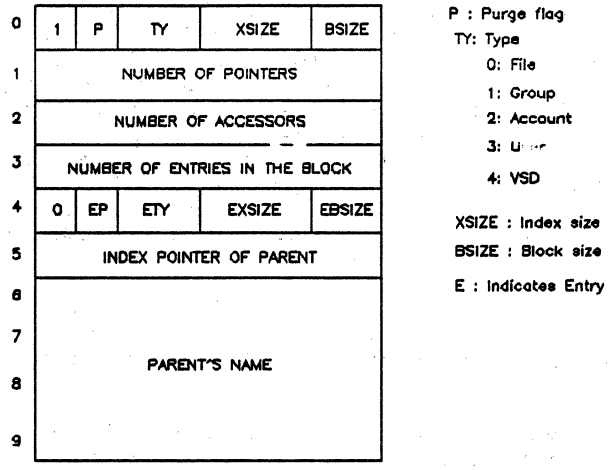


Figure II-2. Index Block Prifix

Index/Entry Block	Entry Size (words)	Block Size (sectors)	Blocking Factor	Unused (words)
Account Index Block	6	3	62	2
Group Index Block	6	1	19	4
User Index Block	6	1	19	4
File Index Block	6	2	41	0
Vol. Set Index Block	6	1	19	4
Account Entry Block	30	3	12	24
Group Entry Block	41	2	6	10
User Entry Block	19	2	13	9
File Entry Block	6	2	42	4
Vol. Set Enrry Block	56	1	2	16

SECTION III: SPACE MANAGEMEMT OF THE DIRECTORY

This section describes the physical layout of the system directory on the system disc. The entire space for the system directory is allocated by reloading the system disc with the cold load media created by SYSDUMP, which contains the directory size.

The system directory space is a contiguous space on the system disc. This area should not have any deleted or reassigned tracks. The sector address of the system directory is stored at the absolute locations 1130 and 1131 (octal) of bank zero memory. The first three sectors of the directory area are allocated for a space bitmap of the

entire area. Every bit in the bitmap represents a sector. For example, the first bit in the first word of the first sector represents the first sector of the directory space area and second bit for the second sector and so on. Sectors are taken from the directory whenever it is needed to construct either an index or entry block. "1" in the bitmap area indicates that the corresponding sector in the directory area is avaiable and "0" indicates the coresponding sector is used.

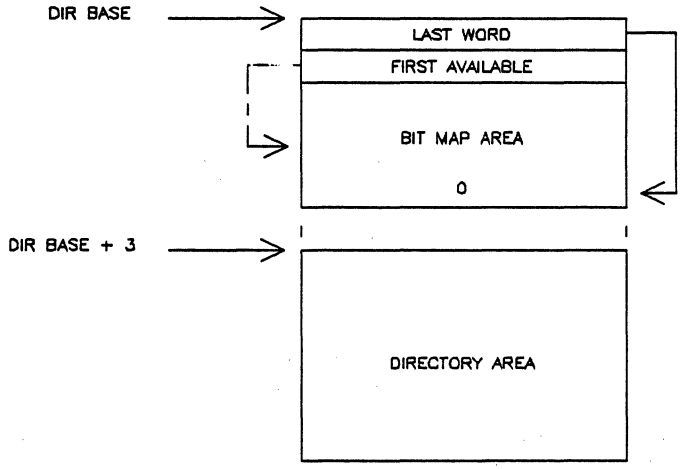


Figure III-1. Directory Space Area

Figure III-1 shows the physical layout of the entire directory area. The first word in the directory area has the relative word address of the last word in the bitmap area. The second word points to a word that should be examined for the availability of the corresponding sectors in the directory area. If any index block or entry block is purged from the directory as a result of purging file, group, user, or account, the space is returned to the directory by setting a proper bit to "1" in the bitmap area. For this reason, the second word cyclically examines the words in the bitmap area to use the new released sectors.

As Figure III-1 shows, the actual system's directory is stored after the bitmap area. Space in the directory area is not preallocated. An integral number of sectors are taken from the directory area to build an index or an entry block. A system with a null directory occupies 17 sectors of the directory area. These 17 sectors are used to build the SYS account with PUB group and MANAGER user in the system.

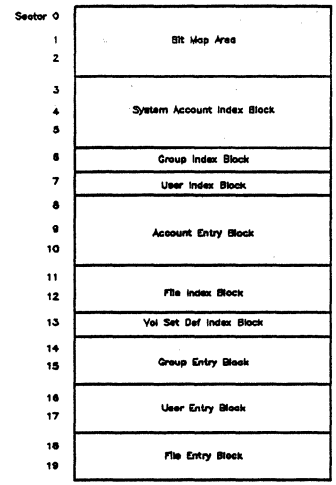


Figure III-2. Directory Layout

Figure III-2 shows the physical layout of the directory after construction of the SYS account in the system. We assume the number of the MPE files in the PUB group is less than 33. As Figure II-2 shows, the index block for volume set definition is allocated even though there is no volume set or volume class defined in the PUB group.

Creation of a new user in an account at the beginning will not allocate any space. The entry will be stored in the entry block that already has been allocated for the first user in the account. This user was created automatically when the account was formed. Adding a new group will use two sectors for file index block and one sector for volume set definition index block. In order to create a new account in the system, there must be at least nine sectors in the directory area to build a group index pointer, user index pointer, file index pointer, volume set definition index pointer, user entry, and group entry blocks.

The MPE's files are stored in the first file entry block of the PUB group in alphabetical order. The maximum number of file entries that can be stored in the file entry block is about 80% of the physical limitation that is shown in Table II-1. When the entry block is utilized 80% by adding a new file in the group, the adjacent file entry block in the group will be examined. If this entry block is less than maximum percent of utilization, the file entry will be stored in that block and two entry blocks are reconstructed to hold equal numbers of the entries. However, if the adjacent entry block is also 80% full, a new entry block will be built and again entries will be spread among the entry blocks. At this time if the file index block is completely full according to

Table II-1, then a message will be sent to the user indicating a directory problem. This algorithm is executed for adding a new user, group, volume set or account in the system.

Two extra data segments are allocated to facilitate the system directory operations. The first is called directory space data segment. The length of this data segment is the size of the bitmap area plus two words. The extra two words are a prefix containing the disc address of the original directory of the system. This data segment is maintained in the virtual memory and used to manipulate the bitmap area. The bitmap portion of the data segment is written to the appropriate directory area whenever the bitmap has been changed by either allocation or deallocation of the directory space. The second data segment is used as the input/output buffer area for all directory management operations, such as insertion and deletion of the entries in the index blocks or entry blocks. This data segment is called directory data segment (DDS).

SECTION IV: SUMMARY AND DIRECTORY CONSIDERATIONS

Considering the basic directory data structure and directory space management, it would be beneficial now to summarize what has been covered in the paper and spell out some of the major directory limitations and directory considerations.

1. Using a two level balanced tree data structure in sub-trees has eliminated a very common problem that exists in the tree type data structures. The problem occurs when the entries are added to the tree in an ascending or descending order. In this case, a tree will become saturated when only half of the tree is full.

2. The size of the system directory is established by reloading the system disc with the cold load media created by SYSDUMP, which contains the altered directory size. The entire system directory area is allocated in the system disc. However, space is not preallocated for any index or entry blocks for accounts, groups, etc. In other words, the following formula documented in Appendix E of the system management reference manual is a very general guideline to detemine the size of the directory desired. This does not prevent the user from adding more entries than the value used in the formula.

   SECTORS=6+6*A+(5.4*G)+(.15*U)+(.05*F)+(.5*VS)+(.5*VS)

3. The total number of accounts in the system, the total number of files in a group as well as the total number of users and groups in an account are all subject to the following limitations:

A. A Physical limitation is based on the maximum number of entries in the index block multiplied by the maximum number of entries in the entry block of each sub-tree. For example, using the table II-1, the file index block can contain up to 41 entries and each file entry block can hold up to 42 entries. Therefore, the maximum number of files that physically fit in a group are 1,722.

B. There is a utilization limit for the entry blocks used in the insertion algorithm. This limitation exists when trying to add entries in alphabetical ascending or descending order and there is no room in the index block and also all entry blocks are at least 80% full. The algorithm will ignor the utilization limit and will allow adding entries to the last or first entry block until it becomes 100% full. However, you will still be able to add entries to other entry blocks by choosing a proper name to cause a proper entry block to become a candidate for insertion of entries.

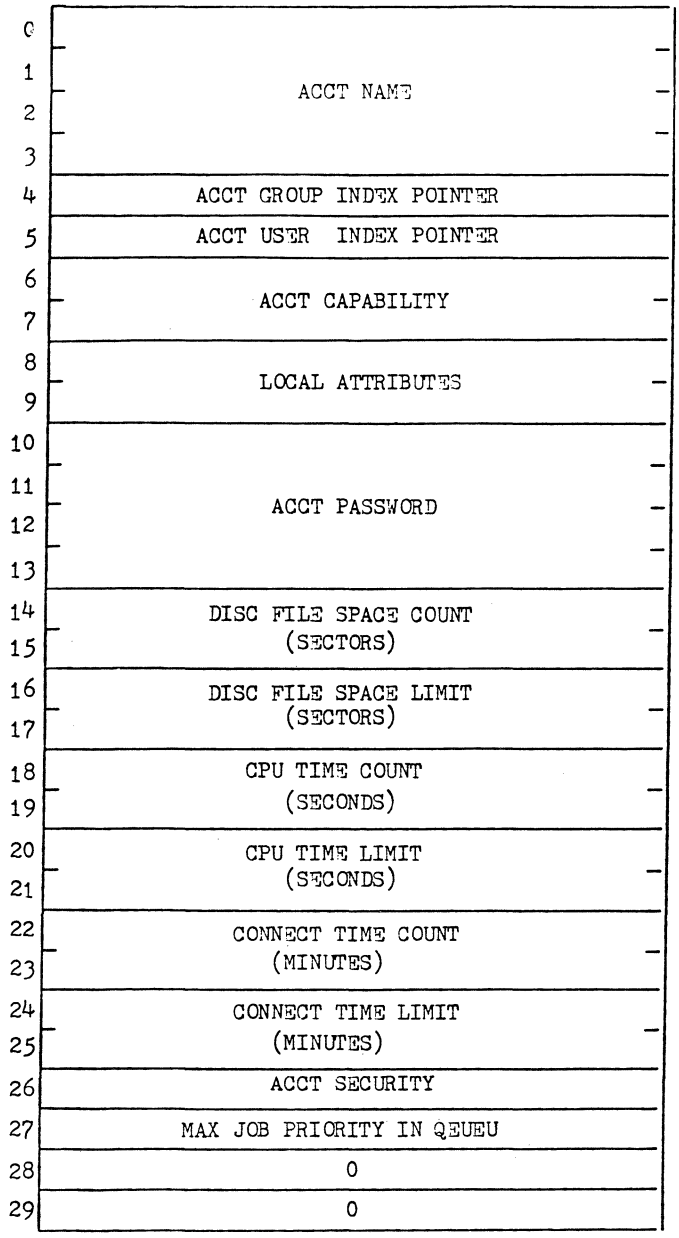
4. There are two major directory performance issues that must be considered:

A. The entries in the index and entry blocks are stored in an ascending order and are searched using a sequential algorithm. If a system has a large number of accounts, groups, and files, it may use more CPU time to locate an entry in the entry block. This scheme coul be replaced by a more efficent search algorithm, such as binary search, to optimize the directory operations.

B. Directory operations require many physical disc I/O operations; in order to locate desired information in the directory, related index and entry blocks must be scanned to get the pointer to the lower level index or entry block. Since access to each block in the directory requires one physical I/O, the operation becomes an I/O bound activity. This will become more evident when trying to access the information in the lower level of the tree or when performing non-logon operations such as using the REPORT and LISTF commands.

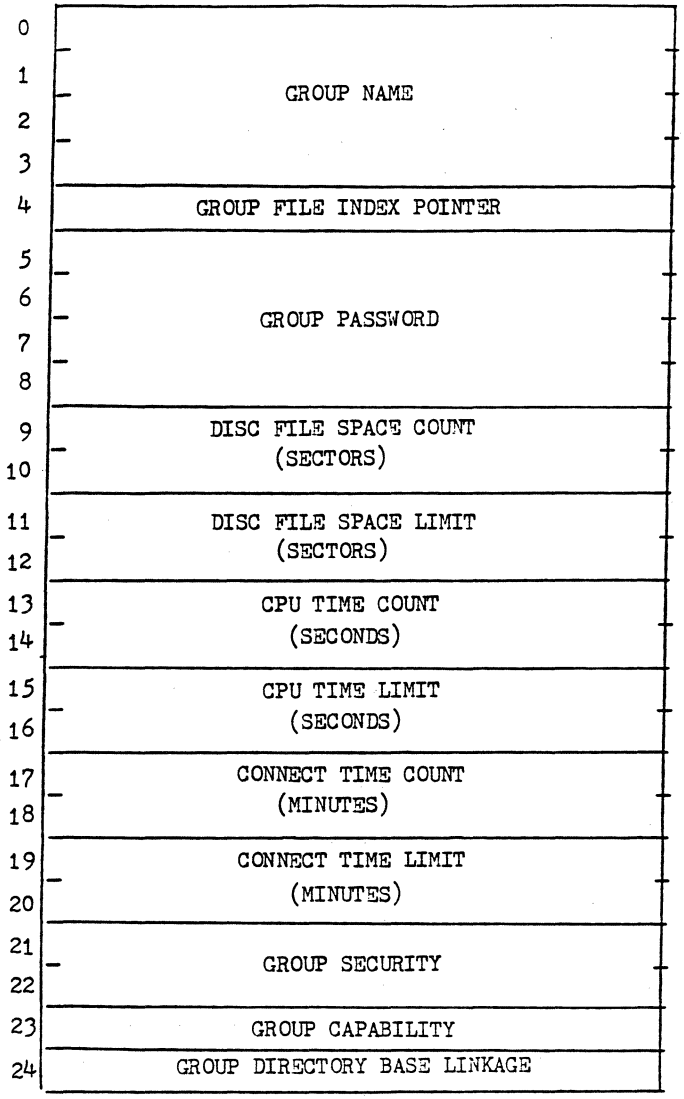
The physical I/O operation can be improved dramatically by using a better physcal layout for the directory or disc casching mechanism for the directory area.

A-1 ACCOUNT ENTRY

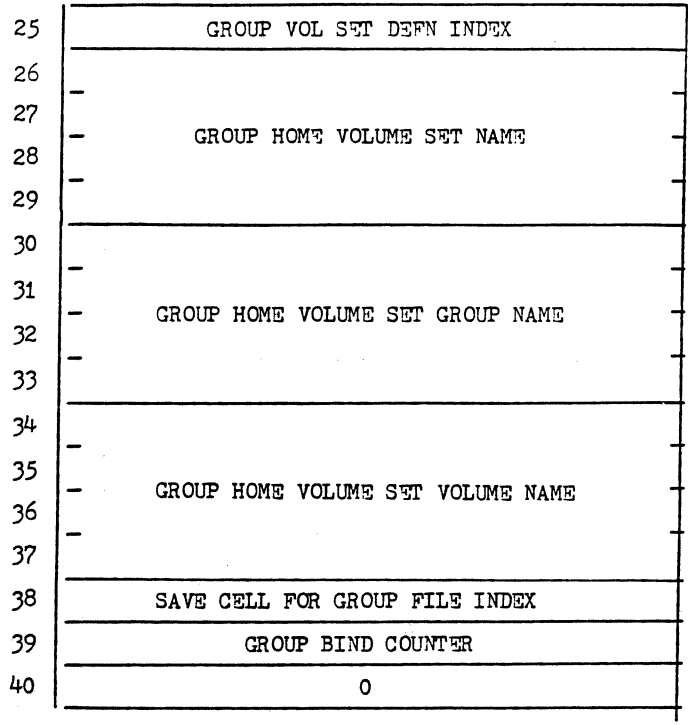


APPENDIX A

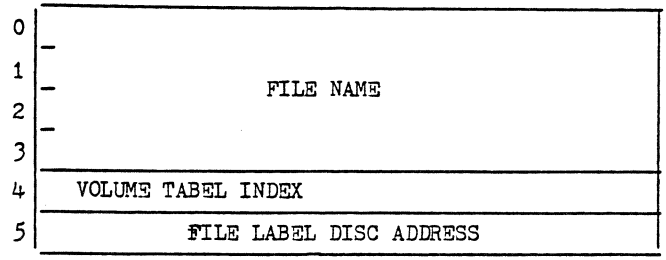
A-2 GROUP ENTRY



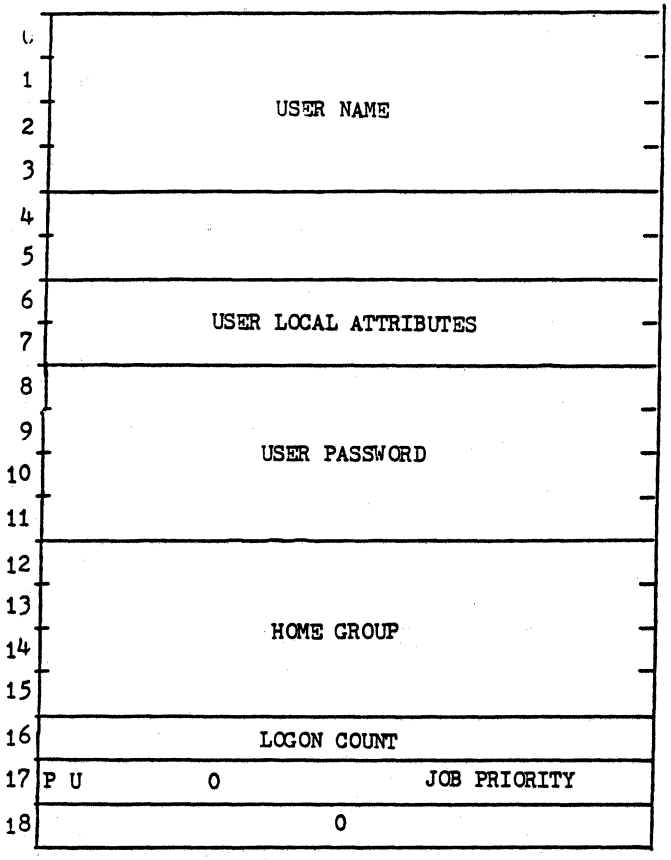
A-3 GROUP ENTRY CONTINUED



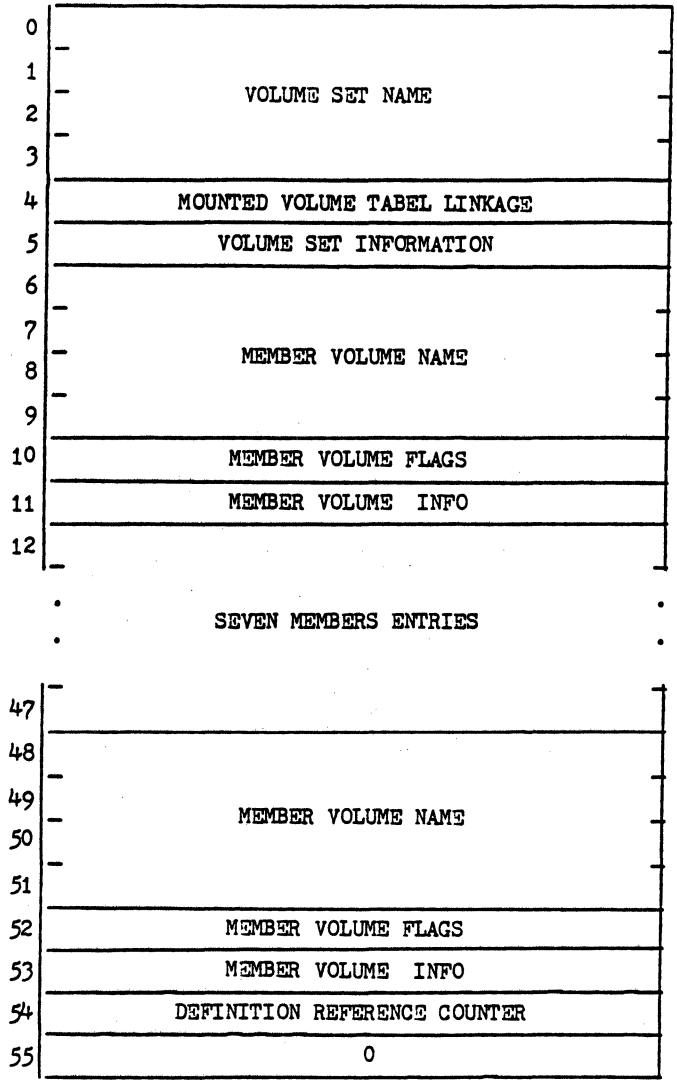
FILE ENTRY



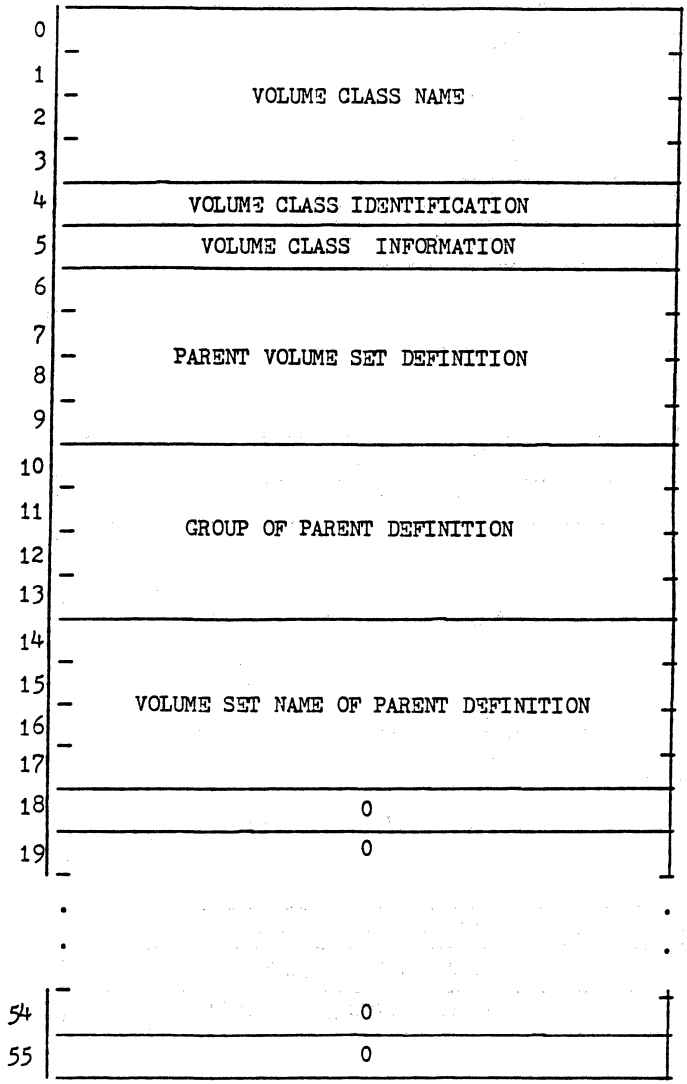
A-4 USER ENTRY



A-5 VOLUME SET DEFINITION ENTRY



A-6 VOLUME SET CLASS ENTRY



Creating Reliable Software Through Automated Testing

Paul Nichols
Hewlett-Packard
Information Networks Division

Software Quality in the 1980's

A major challenge facing the software industry during the 1980's is its ability to deliver high quality software in a timely manner. Many people are concerned that the industry will not be able to meet this challenge. Software quality is composed of many attributes; functionality, usability, reliability, supportability, and performance are all evaluated to determine quality. A renewed emphasis will be place on software reliability as the software industry evolves.

The number of people using computers is increasing dramatically. Some of these people are naive computer users and are less tolerant of software errors than users of the past. In his recent book "Software Engineering Economics", Barry Boehm states that "...by 1985, roughly 40% of the American labor force will be relying on computers and software to do their daily work, without being required to have some knowledge of how computers and software work". Users expect the software to effectively and efficiently help them do their jobs; unfortunately most of us would readily admit that a number of software solutions currently provided are of lesser reliability than we would like. At the same time governments, professional organizations, and unions, recognizing the lack of quality in software, are considering establishing standards for software quality.

Software systems are becoming more complex. Software applications have been created to solve a number of problems and are now being linked together through integration to form broader more complex solutions. The merging of text, data, and graphics into a single output unit is an example of this increased complexity. As software complexity increases the probability of software failure also increases.

Computer systems are being linked together to form networks. A network's availablity is affected by the sum of the failure rates of all the components within it. An increase in the reliability of each component is required to obtain the same reliability goals for a network as measured today for a single system. This is required to achieve today's goals without even considering an improvement in those goals.

The software industry is hard pressed developing and maintaining products and applications to meet the demands placed upon it by customers and end-user departments. These demands result in conflicting priorities for

software developers. On the one hand demands are placed on the industry to quickly develop large numbers of complex software systems. On the other hand the industry is expected to improve the reliability of these systems. Although some very promising work has been done in the area of program proving, it is generally accepted that this type of verification is a long way from having practical application. This means that the software industry is still dependent on testing for assurance that a software product conforms to its reliability goals. This paper examines how software testing tools can make improvements in, and measurements of, software reliability during the testing process while increasing the productivity of the development team.

Software Testing

Testing activities have both static and dynamic aspects. Static testing is performed without actually executing the software, but rather is done by human inspection and peer review of product designs, program code, product documentation, and product training. Static testing is also performed by software tools which analyze the software design definitions and source code structures on the computer. During dynamic testing the software is executed over a wide range of possible inputs and environments for the purpose of finding errors. During the development phase, dynamic testing includes the testing of each individual module all the way through the final software product. During the maintenance phase, dynamic testing includes the retest of a software system in which error corrections and/or enhancements have been made.

Testing is expensive, time consuming and requires large amounts of resources. Even when these resources are committed there is no guarantee that the testing will be done well. The time required to test large systems with a wide range of input values can exceed the useful life of the system. During product development the testing phase requires the largest amount of the programming resource and may represent 40 to 50 percent of the development activity. It is predicted that by 1985 the maintenance phase may consume approximately 80 to 90 percent of programming resources, and a great deal of this time will be dedicated to retest. Experience has shown that committing large resource expenditures can improve software reliability to some extent but still below the level of expected goals. An equally important aspect of testing is the measurement of test results and test progress. Quantitative information is necessary in order to obtain these measurements. Obtaining this information about the testing is often difficult, and the lack of this information may lower productivity and cause the testing to be imprecise. This results in a reduced motivation to find errors by programmers who are not convinced their tests are always exercising the software appropriately, and often they are right.

On the positive side, years of research and volumes of papers, theses, and books have been applied to the topic of testing. Typically, the industry lags behind this research. However, with the proper plans and tools some of this research can be applied today. Some may still consider testing a black art, but with the application of computer science, creativity and automation this pessimistic attitude can be successfully overcome. The keys to success are planning, control, verification and the proper use of tools to automate the process.

There are many elements involved in planning for a testing activity. The test plan should clearly state the testing objectives, the means by which the plan will be executed, and the method of verification to assure proper execution of the plan. A clear description of the appropriate testing process or processes, testing requirements, the tester, the testing tools, and an implementation schedule should also be included. If the testing tools are not currently available the plan should address how they will be obtained. The test plan will vary depending on the type of testing being done. Module testing, system testing, and system retest all present different types of problems to be solved.

Testing Tools

The use of testing tools is an important aspect of software testing. These tools can improve productivity during the testing phase and achieve levels of testing thoroughness not achievable through ad hoc methods.

To derive the most benefit from testing tools they should complement each other. The use of one tool should not exclude the use of another; each individual tool may provide productivity benefits and useful information. When tools are integrated additional leverage from these benefits may be achieved. Tool integration is most easily accomplished when the tools form a tool system. Therefore this should be an important factor in acquiring or developing tools for testing. If the tools available do not complement one another, or if they do not form a complete system, then consideration should be given to modifying them to achieve this.

Figure 1-1 depicts a model of a tool system. This diagram indicates how differences in product diversity affect the tool system. Test generation, maintenance, and measurement tools have the most product specific knowledge and are therefore the most numerous and require the most expertise to use and understand. While test reporting tools have the least product specific knowledge and are the least numerous and should be usable and understandable by individuals that are unfamiliar the the software being tested.

There are other desirable tool attributes which should be considered when creating or acquiring testing tools. If possible a tool should be generalized so that it can be applied to many types of testing processes. It should be easy to use and fit naturally into the way people do their jobs. If extensive training is required for a tool then its training costs may outweigh the productivity increases gained from its use. The tool should be extendable so that as new applications for its use are found it can be easily adapted. Above all, it should be reliable; if not, it may produce inaccurate test results and productivity losses, causing disastrous results.

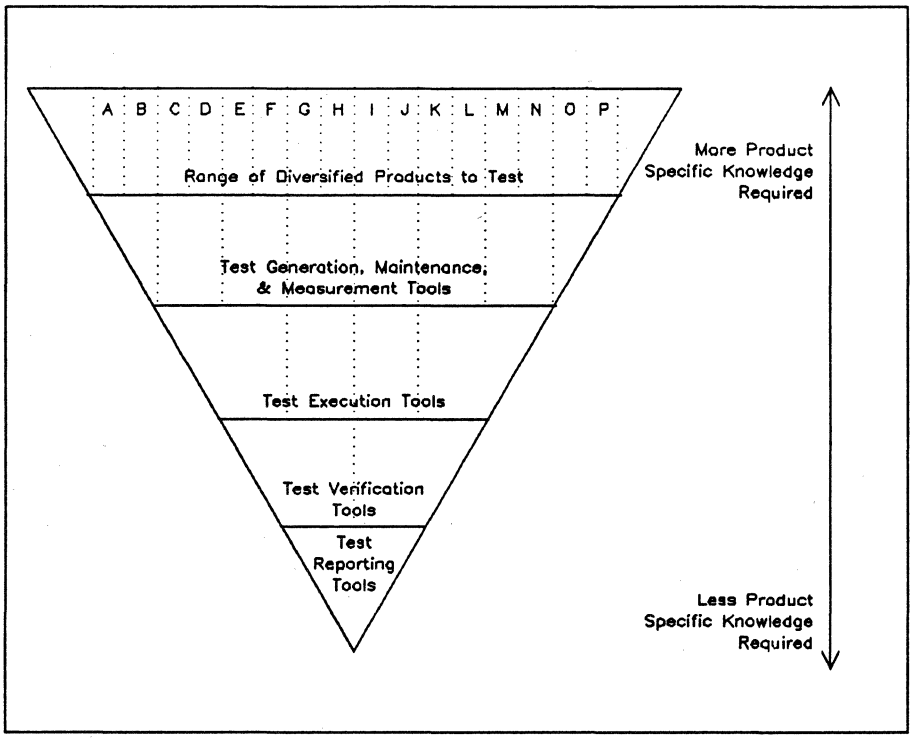


Figure 1-1 Model of Tool Environment

Types of Testing Tools

Testing tools can be placed into categories based on their function. Tools execute tests, collect and report test results, generate test data, organize and manage test cases, and measure test coverage. Some tools will apply to many different types of testing while others will be applicable only to one.

Test case execution tools can have many forms. They may execute test scripts through a simple initiation process or interact with a test in a very complex manner. The initiation of a test script could be as simple as streaming a job, executing a program, or initiating a compilation. On the other hand an execution tool, such as a module driver or stub, may interact very closely with a particular test subject.

Test collection and reporting tools gather test results and translate them into a readable format. These tools present the data in different ways for different purposes. They can report errors, provide feedback on the most productive test cases, and obtain test measurements. This data can be in detail or summary form. The error data should be informative enough for the tester to identify the source of the error so that it can be easily

located and corrected. It should also include information about the number of errors found, as this can be used as a software reliability measure. These reporting tools can also provide performance data as well as reliability data thus giving the tester an early indication of software performance. These reporting mechanisms should be designed to be both programmatically callable and independently executable.

There is a wide spectrum of test data generation tools. In some cases test data can be extracted from an existing file or data base. The tool user may want to subset, modify, and reformat the extracted data through commands. Test data may be saved at the same time as it is manually entered into a program under test. In this way it can be saved for future executions or modified to extend its usefulness. Some very sophisticated test data generation tools can automatically generate test data by evaluating the source code and determining what data will exercise it. Each of these test data generation tools may require different levels of sophistication.

Tools which organize and manage test cases are invaluable. A data base of test information indexed in different ways can pay dividends when a tester is looking for a particular set of tests for a modification just made to a product. Documented test cases, their purposes, results, and measurements can also be valuable when trying to determine the best set of tests to use in a short period of time to obtain the desired results. This organization can also be used to predict the time required for product retest. During the testing phase a buildup of these cases into a organized collection will make the execution of the entire test set developed easier.

The value of tools which collect historical testing information is often underestimated and overlooked. Historical data can serve many purposes, such as showing testing trends, aiding in locating improvements in the testing process, and enabling testors to make predictions about the testing process.

Test coverage measurement tools provide information on the thoroughness of the testing; lack of this information is one of the major problems associated with testing. They are usually very sophisticated tools, being both static and dynamic. That is, they analyze the software source code to determine its characteristics and collect information on its behavior during execution. There are different types of measurements that can be provided by these tools, such as measuring decision statements, non-decision statements, I/O calls, external module calls, and the usage of program variables. These measurements can be provided in detail or summary form and can be targeted toward different audiences. These tools can also be applied to many types of testing processes and can create large improvements in testing productivity by directing testing personnel toward untested areas in the software and providing information on previously tested areas.

There are a large number of tools and ways in which they can be applied. Although tools can be classified according to their function, in actual use tools often apply to multiple testing processes. The application programmer often refers to testing as module testing, system testing, and system retest.

Tool Use During Module Testing

The purpose of module testing is to verify that a module produces the correct output when provided a predetermined set of inputs. Each module should be exercised thoroughly to verify the correct execution of its logic before being integrated into the complete system. There are several methods of module integration; top-down, bottom-up, and sandwich test. If a program is designed from the top down then the modules near the top should be tested first. Conversely, if the bottom layer of modules is designed first then the testing should proceed in a bottom-up fashion. In complex programs it may be necessary to test the modules from both the top and the bottom, hence the term sandwich testing. Using any of these methods avoids what is known as the "big bang" syndrome, whereby all modules are put together at once and tested.

During module testing a module driver may be used to control the module's execution environment. The module driver passes to the module under test a controlled set of inputs and receives back the output results. A module driver may be very complex and have detailed knowledge of the proper input types and values required by a module. The module driver may verify the outputs passed back by the module under test, or call another tool which does the verification, or simply store the inputs and outputs passed between the driver and the module for later verification. A module stub may be required when the module under test calls a module which is not part of the test scenario or which does not yet exist. In this case the function provided by, or the data requested from, the missing module must be simulated by a module stub. The module stub must produce known or predictable results. Under complex situations producing predictable results can be difficult; in these cases it is better to use a previously tested module rather than a stub. The use of bottom-up testing may allow a module to use lower level modules, which have already been tested, in place of a stub module.

Verification of proper module execution can be accomplished through the use of verification tools. The tester should establish in advance the expected execution results for each set of inputs used. The verification tool can then verify the outputs provided by the module under test against these expected results. In some cases the output produced by the module under test may be written to a file or a data base. In these cases a file comparison program can be used to check the test results against the expected results.

Test coverage measurement tools are used to collect information about the execution behavior of a module during testing. Decision coverage analysis, statement coverage analysis, data usage, I/O calls, and external module call information can be used by the tester to determine if enough testing coverage has been achieved. These execution measurements provide the tester with information required to locate the untested areas in the module. This information can also be used for early performance tuning of the module.

Tool Use During System Test

The purpose of system testing is to comprehensively exercise the complete software system to find errors, while determining that it produces correct

results. An additional objective of this testing should be to produce a comprehensive set of automated test cases which can be used for system retest.

The test data generation tools provide an important means of quickly creating test data for this phase in product development. These tools may take data from a similar software system, modify and reformat it to produce suitable data for input to the new system. In an interactive system, the data entered can be saved along with the output results to produce reusable test cases.

The test cases, while running, can report on their progress, status, and error detection in an automated fashion. This requires that the test cases be self-checking and communicate with some type of test reporting tools. This reporting is directed to a test log or data base associated with the test cases being run. The test reporting tools can denote the start and completion for an individual test and test series. This is useful to determine if a particular test case or series of test cases executed to completion or aborted for some reason. They can also provide check point information for long tests. This is convenient when trying to trace an error back to its origin. The tools can also provide test case documentation by placing descriptive information into the test log. Summary as well as detail information is useful in test case reporting tools. These tools are an important means of providing automated test result reporting.

Test case organization tools are useful during this testing phase. Typically during system testing large numbers of test cases are being created. As the number of test cases increases it becomes more difficult to manage them in an organized fashion. These tools help testors manage the large number of test cases being created. As test cases are completed the tester enters the test case and its relevant information into a data base accessed by this tool. This data base identifies the nature and purpose of each test case along with execution and verification information. This information is used to assist a tester in locating a set of tests for a particular function. As the software integration continues, this information can be used to re-execute and re-verify the tests for the latest version of the software. The tests may be organized in a number of ways. They can be organized by the function or functions they test, how successful they were at finding errors, or in some other logical hierarchy. Whichever way they are organized the tool should provide an easy way for a tester to add new tests to the test data base and to determine the proper set of tests to run.

Measurement tools play an important role during system testing. Because large numbers of tests are often needed to fully test a software system, the measurement tools can direct the tester toward areas of the software that have not yet been tested thus enabling the tester to create test cases which represent a comprehensive test set for the product. In very large systems it may be necessary to take measurements of smaller pieces of the system and accumulate the measurements in order to obtain a complete picture of the testing coverage. Normally, there is a significant amount of overhead associated with using these tools. For this reason these measurements should not be taken on a continous basis but rather be taken at intervals during the testing phase.

Tool Use During System Retest

The purpose of system retest is to assure that the changes made to the software have not caused its reliability to regress. After enhancements have been added to the product, or error corrections made for problems found, the product is retested. New tests written to verify proper execution of the enhancements, and corrections made to the software are then added to automated test data base. This updated test bed is then re-executed to verify that the product still conforms to its reliability goals. This type of testing is typically referred to as regression testsing.

Many of the same testing tools used during product development are useful again during system retest. The test organization tools are used to add the new tests to the data base. The test execution tools are used to re-execute the test. The test verification tools will allow the tester to quickly verify the test results and determine if they were successfully executed. The measurement tools will assure adequate coverage for the new enhancements.

Automated Testing Experiences

Hewlett-Packard has a long standing commitment to protect our customers' HP-3000 software investment through upward compatability. In addition, Hewlett-Packard has a commitment to customers that the integrated HP-3000 product line software releases will be installed by the field, minimizing the need for systems programmers at the customer site. These two strategies have resulted in a significant amount of work in integration and testing for Hewlett-Packard. Inevitably, this investment in people and time has encouraged us to consider ways to improve our productivity as well as the reliability of our products.

The ever-increasing number of products which run on the HP-3000, combined with our decentralized organizational structure, provides challenges for testing and product certification which require automated, transferable solutions. For these reasons testing tools are used by Hewlett-Packard to automate the testing process on the HP-3000. These tools make possible the creation, execution and verification of automated test packages, thus enabling divisions to test product revisions more quickly and accurately and to improve the testing process during new product development.

This effort was started by developing file and data base comparison tools for verification, and a test logging facility for automated reporting. It also included using self-checking test cases and in some cases modifying products to allow automatic input of the test data. The file and data base comparison tools have flexible features for generalized use. A variety of file types and data sets can be compared and the data comparisons can be made on a conditional basis. Additionally the comparison tools can report their results directly to the test logging facility. The test logging facility uses a standardized report format and can be called programmatically or run stand-alone.

These efforts allowed test automation for a number of products. However, test automation for all products could not be accomplished with only this small set of tools. Some unique products, such as IML/3000, required still

other specific tools to automate the testing process. In these cases special purpose tools were developed to aid in test automation for that product [Marcos82].

The creation of automated test packages for interactive products required the development of a special tool system. This system saves the input entered into the product by the tester and the resultant output from the product as a test case. These test cases can be "replayed" automatically and the results verified and reported.

The testing of an entire system, which encompasses running all of the automated test packages for various system configurations, required the development of a test monitor facility. The test monitor facility, known as the "test umbrella", performs computer operator functions, manages the overall testing for the system, and has the ability to detect unique errors in a system or product that an individual automated test package is unable to detect.

Many of the same tools which allow the creation of automated test packages are also used in testing during new product development. An additional tool, called a decision path analyzer, is used to assist the tester in finding untested areas in the product and obtain a measure of testing coverage achieved by the final test package. Experience with this path analysis tool indicates that the tester can expect the first set of test cases to obtain a 40% to 60% coverage measure. The coverage should then increase quickly until an 80% to 90% coverage is achieved. At this point, increasing the testing coverage may become very difficult and in some cases not worth pursuing, depending on the nature of the untested code.

The testing tools used to automate and quantify the testing activity have proved to be invaluble. They have enabled Hewlett-Packard to make improvements in the testing process which would have been difficult, if not impossible to do without them. The automated test packages make testing much more versatile and accurate and allow configuration testing to be decentralized. The quantifiable information enhances the accuracy and productivity of the testing process. Today there are approximately 50 automated test packages used by divisions to verify product reliability on the HP-3000. Figure 1-2 illustrates how a single automated test package can be used by many divisions to perform configuration testing.

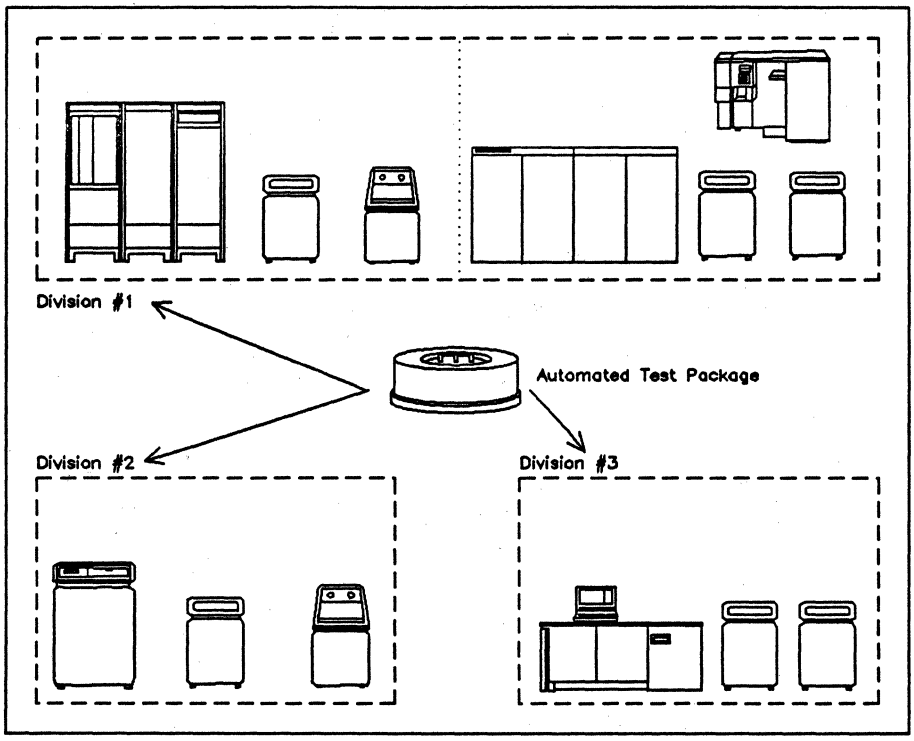


Figure 1-2 Decentralized Configuration Testing

Cautions Concerning Testing Tools

Care must be taken when deciding what tools to use and how to use them. A software tool, like any other tool, merely extends human capabilities. When the proper tool is used benefits can be gained from its use, but when the wrong tool is chosen or used improperly benefits may not necessarily be achieved.

There are a large number of commercially available testing tools today. Acquiring a tool that precisely meets the needs of a particular testing environment may be difficult. Care must be exercised before acquiring testing tools. Acquiring testing tools should be approached as carefuly as with any other software purchase. The tool must fit your needs. It should be of good design, well documented, reliable, and supported. If the tool is complicated to use then training should be available to assist in its use. Probably the best way of evaluating such tools is to ask the vendor to provide a list of users and talk to them about their experiences.

Early identification of the testing tools needed for a project is important. It is important to make sure the tool will be available before its use is

required. If the decision to develop a needed tool has been made, make sure the tool's design provides for extended use after the project has been completed. Also, consider designing it with enough functionality so that the tool benefits can be shared by other projects. Provide for tool support within your development environment, through documentation, training, and technical support.

In some cases, testing tools may require changes to the way products are developed. This may mean altering the development process or even the products themselves. Making changes to the process must be communicated clearly to those affected and seen as a positive improvement, otherwise resistance will occur. Software under development may require "hooks" to accommodate some tools. These modifications should be designed and implemented to be transparent to the software users.

Conclusion

In the future, reliability will be an even more significant factor in software systems. As software systems become larger and more complex it will be difficult to guarantee this software attribute. At this time, testing is the best method known to assure reliability. This requires a rigorous planning method and a consistent application of the plan to all phases of the testing process. Testing tools provide the leverage needed to achieve increases in reliability and at the same time increase the productivity of the testing activity. These tools may have a variety of functions and can be applied to many testing processes. The largest benefits are realized when these tools are integrated and complement one another. Care must be exercised when acquiring and using tools. In the future improved testing tools and techniques will be discovered. As these new technologies become available Hewlett-Packard will continue to utilize them to develop, evaluate, and automate the testing process, thus improving our ability to assure reliable software systems.

Bibliography

[Miller81] Miller, Edward.,Howden, William E. "Tutorial:Software Testing & Validation Techniques", Second Edition, 1981. Published by IEEE Computer Society Press.

[Myers76] Myers, Glenford J., "Software Reliability Principles & Parctices", John Wiley & Sons, New York, 1976.

[Boehm81] Boehm, Barry W., "Software Engineering Economics", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[Schindler81] Schindler, Max, "Today's Software Tools Point To Tomorrow's Tool Systems", Electronic Design, July 23, 1981,Vol. 29,Number 15.

[Marcos82] Marcos, Gary, "Practicality in the Design of Software Testing Tools",Hewlett-Packard,Information Networks Division,1982.

'User-Friendly' REALLY Can Be Friendly -- If Finding The Answers Isn't A Hassle

by Sylvia Onalfo-Wybrant
Technical Communications Manager
ASK Computer Systems, Inc.

Even the most friendly, user-oriented computer or software application system can be difficult to access if finding answers is such a time-consuming, frustrating process that it inhibits the user's desire to ask questions.

What makes documentation ultimately "work" is not only that complete answers are provided, but that the quest for information is inviting, helpful, and satisfying. Certain documentation quests, most notably via on-line tutorials, may even be fun.

Passing references in technical or computer publications routinely chide technical communicators for not presenting information in a user-friendly manner. Not surprisingly, microcomputer publications are at the vanguard pushing for an easy-to-reference style of writing for their vast first-time user market.

Cary Lu provides this interesting analysis of documentation in a recent High Technology magazine article [1]:

> The first line of support, the written manuals, varies in quality from bad to abysmal. With recent progress, a few manuals are now mediocre. Too many 500-page manuals have no index, much less a comprehensive, cross-referenced one...good documentation is so rare that a case can be made for selecting products on that basis alone.
>
> Since the computer itself is a powerful teaching tool, a few products have interactive training software, an excellent idea that sometimes works. No one has carried the idea beyond the simplest steps, though.

This negative assessment of current documentation is even more significant when you consider these statistics provided by the International Association of Business Communicators (IABC) [2]:

- The production, processing, and distribution of information comprises nearly one-half of the GNP of the U.S.
- Thirty-one billion original documents and 630 billion pieces of mail were generated by (U.S.) businesses in 1980.
- By the year 1990, nearly half of the nation's work force will be employed in the creation, transmission, and management of the copious information flow. The compensation costs for office managers alone is expected to approximate $1.35 trillion.

Despite these staggering projected dollars and word counts -- and the consensus currently in vogue that state of the art technical writing should be user-friendly -- little has been documented about how to produce "friendly" communication.

Technical writing is neither simple nor unnecessarily complex writing; nor is it a mystical process. It simply requires hard work, good organizational and interviewing techniques, and a predeliction for accurately portraying details. Technical documentation shares a common thread with such seemingly diverse writing forms as the poem, the play, or the essay. All address broad subjects in an obvious and concise manner. All are most valued when there is a sense of clarity and crispness evident in the writing.

A promotional ad for Personal Computing magazine, subtitled "There may be a latent Hemingway, Fitzgerald, or Asimov beneath that Einsteinian exterior," offers this advice regarding good technical writing [3]:

> Keep your writing simple...Don't clutter your piece with unnecessary jargon...Feel free to use 'I' and 'you' to make your article more personal and meaningful to the reader...(and) make sure your details are accurate.

Remember the basics

Readers and writers alike should understand two basic documentation realities:

1. Even the best of user reference documents cannot, and will not, easily answer all user questions. Like written news reports, technical material will only reflect changes up to the arbitrary production deadline.

2. There is an unspoken, probably subconscious, rule that any user looking for reference information will first crack open the manual or otherwise look for written help at the last possible moment it's needed.

Certainly people who are in a problem-solving mode appreciate direct answers; but we also understand that, depending on the problem's level of complexity or inflexiblity, simple answers may not be enough. What we are looking for in documentation then is a roadmap to help us find our own answers.

Technical documentation can be easy to read if a very few guidelines are stringently followed:

- Realize that anyone looking at documentation for the first time, despite credentials or knowledge level, is a first-time user.
- Organize for the first-time user.
- Keep it simple.
- Keep it visual.

For documentation to be effective, the documenter must have designed the reference work with two basic concepts in mind:

AUDIENCE
and PRESENTATION.

WRITING doesn't need to be a hassle

If you're directly responsible for documenting a technical subject, there are some guidelines you can follow to be successful. These include:

- Write simply.
  This means being concise and precise. Especially with beginning drafts, write about a concept the way you would orally talk about

it. Avoid redundant expressions, but if you MUST repeat information, be consistent; or explain an idea once and then cross-reference if necessary. Break different concepts into smaller sentences to aid the reader's retention. Addressing your audience directly is very user-friendly, as in this example: "you should use the present tense whenever possible." You also may wish to consider using implied subjects such as the implied "you" in: "use the present tense whenever possible."

- **Edit relentlessly.**
  As previously quoted, Personal Computing makes the recommendation to prospective writers to "keep your writing simple." Even this sentence can be edited to be "write simply."
- **Organize and revise.**
  The magazine goes on to outline how to organize an article about a particular program:
  - o state the purpose;
  - o show a sample;
  - o explain the options;
  - o show another example;
  - o explain the underlying theory;
  - o analyze the details;
  - o suggest how the reader might improve or change.

  Starting with a topic outline, expanding it with sentences, and then adding paragraphs and supplemental explanation is a good technique for defining priorities and keeping an overview of an entire project.
- **Make presentation as important as information.**
  Format sentences as visual blocks and go out of your way to be graphic by including bullets, indentations, examples, flow charts, matrices - whatever will enhance readability. But avoid being graphic for graphic's sake. Learn and use your editor's formatting capabilities. Highlight, underline, or bold concepts to make screen information as visual as possible.
- **Provide what the user wants.**
  Be sure "user-friendly" really reflects the user's point of view and not some vague or preconceived notion a writer or engineer, however well qualified or credible, might have of what the user wants. Assertive thinking (giving yourself permission to talk to customers) and hard work (actually setting

aside time for personal contact and then making contact) are the keys here.

- **Use examples.**
  At ASK Computer Systems, we actually created a simulated manufacturing company data base to provide customers with examples of how the programs would work in a quasi-realistic environment. Customer feedback has been favorable because the examples show not only how the programs function, but provide possible application uses as well.
- **Include negative information.**
  The typical stance regarding error messages and major system limitations is that they are not inherently "user-friendly" and if documented may detract from the product's salability. Also, they typically are finished at the end of the R & D cycle; and they usually are boring. Consequently, these references are generally excluded from most documentation. Technical communicator Christine Browning makes a strong case that this should not be so [4]:

  > A teacher once asked this question of a group of software writers:
  >
  > 'What is the most important part of a software reference manual?'
  >
  > Nobody got the right answer. Section 1 received the largest number of votes; the index came in a close second.
  >
  > According to this teacher, the section containing the error messages was the most important. Why? Because if you have an error, you are finished unless the manual explains the error and tells you exactly how to recover.

- **Qualify whenever possible, but don't quantify unless you need to.**
  Use dashes, bullets, indentations, and other graphic devices to break up copy, but use numbers only when necessary since the reader's tendency is to rank information when it is put into numerical order. Also, formats using numbers are more difficult to update since what is currently number one may be preempted in the future and the descending numbers will need to be changed accordingly.

- **Be a jargon-watcher.** Computerese can be like legalese - hard to resist and harder to comprehend. Technical communicator Ken McGinty pokes fun at the "ese" problem in an article entitled "Bureaucratese: The State of the Art" [5]:

> ...bureaucratese is thriving...Redundancy continues to be strong in government language...(as in) real truth [like the true facts, I guess], practical usefulness, past history, especially unique opportunities, and desired objectives [as opposed to the undesired ones?]...
>
> (Additionally,) bureaucrats seem to live in different dimensions from real people...Nothing ever happens now; it happens at the present time or at this or that point in time...

READING should be an active pursuit

While reading generally is regarded as a passive pursuit, you can make it an active and more satisfying experience by assuming certain responsibilities. For instance:

- **Anticipate the audience type the writer is addressing.** This will provide a benchmark for orienting your expectations or a roadmap for finding difficult answers. If the writing style is tutorial, you are probably expected to be a beginning user; if sentences are presented in a more complex, less how-to manner, a certain prerequisite skill or interest level probably is assumed.
- **Get a "feel" for a document before you begin reading or referencing it.** A document's organizational style may inhibit or enhance a reader's ability to retain information. Good documentation -- optimally -- should flow logically and be easy to reference; but deadlines and equipment limitations, and yes even the writer's conceptual inadequacies, often result in less-than-optimal documentation being available. Again, you can orient your expectations -- and increase your satisfaction -- if you become familiar with how the document is organized before you absolutely need to find an answer from it.

APPENDING means making it right for you

If you're at the mercy of technical reference works that "don't quite work," consider doing what manufacturers and technical support groups have been doing for years: make it right for you. The basic guideline here is to update quickly -- as soon after reviewing the original documentation as possible.

An in-house document can be a major "procedure book," a minor rewrite of specific sections, or a graphic addition that makes a complex concept comprehensible. It can be supplemental information such as an index, error message listing, or glossary of terms.

When appending, be aware that there is a delicate balance between waiting too long and not doing enough soon enough. Timeliness is as important as accuracy -- and while wishing to create THE MOST PERFECT copy is a nice goal, it is probably unattainable; the opposite pole is to waste time and resources trying to limp along using something that's inherently not capable of doing the job.

Feedback is your responsibility too

When either READING or APPENDING, be sure to be in active mode and send feedback back to the original writers or designers. If you don't think the documentation is complete enough, let the writers or designers know how you feel. They also enjoy knowing what really helps.

Don't turn the writing of your comments into a major task; a simple note or outline is better received than an elaborate intention that is never fulfilled. Or, if you produce a generalized revision to your vendor's existing documentation -- that is not proprietary to your company -- consider sharing your revision with the vendor for inclusion in their next documentation update.

Know the audience's needs

Understanding the audience is an important factor when documenting. Writing style and presentation build upon the foundation of who is being addressed. IABC communicator Jim Dodge makes this point about audiences [6]:

> Sometimes the business of communicating is a lot like whistling in the dark. You suspect

> you have an audience, but you don't know how large it is, what it looks like, or if it's enjoying the performance...(demographics is the) key to understanding what makes that audience tick...for communicators it (the audience) brings our whistling into the light.

He goes on to state an easily forgotten concept:

> One factor that complicates developing a better understanding of audiences is the rate at which they change.

The best writing is direct writing and knowing what users want expands the writer's vision. Too often, especially when documentation is being written for products still being developed, users are contacted only at the end of the process -- if at all. In such a scenario, the writer stands the chance that the audience may have changed.

The best way to include an abundance of applications-oriented information, even for the most technical and abstract subjects, is to have direct user input throughout the documenting process. At ASK Computer Systems, for example, we are instituting a formal "user interface" program to assure writers the opportunity for contact with users. This program will help ASK writers to:

- -- Use internal customer documentation and actual phone or on-site interviews as reference tools.
- -- Call users directly if at all possible.
- -- Work with the R & D, education, sales, and other staff in our organization who have different perspectives regarding customer wishes.
- -- Survey users before and after a product is documented.

Other companies expand upon this approach. A large Silicon Valley micro manufacturer solicits approximately 50 users to come in-house and test documentation before product release, placing as much emphasis on the ability to understand the reference tools as on the product itself. At a recent Society of Technical Communicators meeting, one HP technical documentation manager explained that his group internally and systematically pretests documentation by requiring HP personnel unfamiliar with a project to try to operate technical systems before the product is released.

Interfacing with Subject Matter Experts

Good "friendly" writing reflects the writer's clear view of the subject. For this to occur, there must exist a good liaison between the writer and the subject matter expert and/or the subject.

There are successful techniques that can be used to create or enhance liaison (and more credibility) with "more technical type" information sources. These include:

- Good interviewing techniques are a must. Be direct, courteous, and friendly when interviewing.
- A fill-in-the-blanks approach is an easy way to solicit initial information. A resource person provided with a draft, which may include questions to answer or blanks to "fill in," approaches the task of reviewing the document with more of a "team" spirit than as an individualistic "writing" endeavor.
- If at all possible, the writer should use the product -- preferably as a first-time user.
- Don't ever be afraid to ask for clarification, no matter how trivial your questions seem.
- Consider tape recording your sessions. The benefits are numerous: your source doesn't need to slow down for you, you won't miss a single comment; and you can recreate the interview at any time convenient for you.

Presentation

Presentation is important no matter how dry the information or how, supposedly at least, technically oriented the user is. The type of user-friendly presentation that can be developed depends upon the type of documentation. The following two documentation types lend themselves to the user-friendly approach.

-- **Manuals** are the mainstay of technical writing and typically include so much specific information that the overview often is lost. Indexes and flow charts help. Graphic plotters can provide an easy way to produce professional visuals. Or, you may have in-house graphics

personnel who can lend a hand to make difficult concepts visual.

-- On-line computer documentation is undoubtedly the most "user-friendly," but also is the most format-bound of all documentation types due to the size limitations of the standard screen. Text editors often are restricted in their ability to visually highlight information so that it isn't dry looking.

Technology will become increasingly screen-oriented -- a consideration most writers and designers should be preparing for now. This means "writing visually" as well as simply. Audiovisual and educational writers, as well as scriptwriters, have developed skills for visual writing that should be referenced and used.

Speculating on the future of the Information Age, Stanley Marcus, co-founder of the Neiman-Marcus department stores and author of the book Quest for the Best, says [7]:

> The communicator is going to have to make much greater use of the electronic tape or whatever comes along to replace it eventually, rather than paper, in disseminating information across the country.

He goes on to add this aside:

> I find some objections to this. You don't have a chance to think as much when you look at tape...You can argue with a book, but you can't argue with a television screen. But I think as a communication device it is going to supersede all others.

In Conclusion, A Little Perspective

Lest you feel that by following all these suggestions you will master the technical writing dilemma forever -- or now feel so overwhelmed that you don't even want to try making things simpler -- a closing quote on the future may provide some perspective.

Dr. Arno Penzias, winner of the 1978 Nobel Prize in physics and research executive director at the

Communications Sciences Division of Bell Laboratories, speculates [8]:

> If given a choice, people are likely to choose those machines that allow them to communicate in the same way they communicate with other people. They will choose machines and services that have friendly interfaces...
>
> **Perhaps writing is a temporary aberration of human history caused by the limitations of technology.** It is much easier to scratch symbols in the mud with a stick than to build a system that can recognize spoken words and convert them to test. But that does not mean that writing is more fundamentally suited to human needs than speech recognition. In an era of technological choice, transfer of information -- whether text, graphics, sound, or video -- will more closely approximate human-to-human interactions...
>
> ...Computer users will act as computer designers, making use of one kind of friendly interface to help create others.

Until this visionary time comes, however, writing simply and clearly is, and will remain, state of the art.

REFERENCES

[1.] Cary Lu, "Microcomputers: The Second Wave," High Technology, September/October 1982.

[2.] James Woods, Jr., "Frentic, Automated Future Awaits Communicators," IABC News, November 1981.

[3.] Staff, "How To Write For Personal Computing," Personal Computing, November 1978.

[4.] Christine Browning, "One Teacher's Opinion," CALLOUT (newsletter of California's El Camino chapter of the Society of Technical Communicators), September 1982.

[5.] Ken McGinty, "Bureaucratese: The State of The Art," Rough Draft (newsletter of the Phoenix chapter of the Society of Technical Communicators), March 1979.

[6.] Jim Dodge, "Lifestyles and Demographics," Communique (newsletter of the San Francisco International Association of Business Communicators), January 1983.

[7.] John N. Bailey, Executive Director, "The Challenge: Communicating in the 1980s," Journal of Organizational Communication (magazine of the International Association of Business Communicators), 1980/2.

[8.] John N. Bailey, Executive Director, "Technology Will Accomodate People," Journal of Organizational Communication (magazine of the International Association of Business Communicators), 1981/3.

ADDENDUM

Sylvia Onalfo-Wybrant has been Technical Communications Manager for ASK Computer Systems, Inc. for the past three-plus years. ASK is a young, growing software firm providing computer systems for manufacturing firms. Sylvia also writes audiovisual scripts and presentation treatments on a freelance basis. Previously, she was corporate communications editor for a multi-service utility company located in San Francisco, held editorial and management positions for the fur and food industries, and spent five years working on Southern California newspapers as a reporter and editor. She recently was awarded the M.A. degree in Playwriting from San Francisco State University.

IPC Files : Why Haven't You Used Them Yet?

by Harold PARNIGONI
PARANGON Enterprises
62 Blvd Du Domaine, Ile Perrot
Quebec, Canada, J7V 7P2
(514) 453-9243

INTRODUCTION

How many times have you used the TELL command to send messages to other users on your HP3000 system and said, "Gee, if I could send messages between programs I could make some really neat applications." Probably never (at least not in those words). But the ability of two processes in the same job/session to communicate allows for a greater deal of co-ordination and sophistication in programs. The MAILBOX intrinsics do provide this kind of communication. But it isn't very reliable or easy to use. And the MAILBOX intrinsics do not allow for communication between two different job/sessions much less communication between two processes on different machines. IPC (Inter-Process Communication) files are an easy and elegant method for processes to communicate.

Basic Conceptual Description

MESSAGE FILES :

Message files are queues or First-In-First-Out (FIFO) data structures. The first message written to the file is the first one read. As a queue there are two ways to access the file. Either you are a reader or a writer. Writers add to the end of the queue and readers take from the beginning of the queue.

Think of it as a tube. Writers are at one end, readers are at the other. Writers place the messages, one at a time, into the tube opening at their end, and readers take the messages out, one at a time, at the other end. Notice that you cannot read the messages in any order other than the order in which the writers placed the messages into the tube. You cannot read the third message in the tube unless you have taken out the first two messages. Logically you cannot be a reader and writer at the same time. You have to run from one end to the other changing from a reader (writer) to a writer (reader).

This model applies to message files very well. When you open the file you must indicate if you are opening as a reader or a writer. To be able to read and write you must open the file twice: one as a reader, another as a writer. Applying this to our model, you are hiring two guys, one at each end of the tube, who communicate with you via telephone.

The model applies in message reading and writing. When writing it's simply an add to the end of file (end of tube). If the file is full (tube is full) you have to wait or an error condition will occur. Which will happen will depend on the mode of write (We will discuss the details later). For readers the model demonstrates

the major difference between regular file and message files. To read the second message in the tube, you must take out the first message. Message files accomplish this by destructive reads. As you read the file, each message read is taken out of the file. The next message in the queue comes to the front. This happens no matter which of the many readers take the message. Non-desructive read mode is available, but it does not move you into the queue. It's like reading the message at the front on the tube without taking it out. But you still can't read past it to the next message. You must take out the first message to get at the next.

The "TUBE" model shows how a user perceives messages files. In reality it is nothing more than a serial file. But the file system intrinsics (FOPEN, FCLOSE, FREAD, FWRITE, etc.) treat files of type message as our model describes. The file label contains the information on readers and writers, the pointers to top of queue and end of queue as well as file size, etc. Because the file system knows exactly how to treat the file to make it look like a queue we don't have to bother doing the checks. For this reason, message files are much easier to use than creating your own serial file and figuring out all the exceptional conditions.

Accessing Message Files

The beauty of message files is that access is through common file intrinsics. Some intrinsics cannot be used on message files. Logically FUPDATE, FPOINT, FREADSEEK, FDELETE, FREADDIR, FWRITEDIR and FSPACE cannot be used since they imply being able to reorder the messages in the queue. All of these intrinsics will fail when used on message files.

To identify a message file is quite simple. On a :LISTF,2 look at the TYP field. If the character under the "P" is "M" then it's a message files. To build one is just as simple. In the :BUILD command add ;MSG.

Below is brief description of each intrinsics and how they can effect message files.

FOPEN

This is where you open access to your message file. In opening a file you describe several important items.

- Access type : by opening up in READ access you become a reader. If you are the first writer to open up and you open up using WRITE access it will cause the clearing out of the file. If you are second writer to open your access is automatically altered to APPEND access.

- Exclusive : setting to SHARE will allow for multiple readers and writers. In SEMI access only one reader is allowed but there can be many writers. The default is EXCLUSIVE which is one reader and one writer. Note that if you wanted to lock the file completely you will have to open it in EXCLUSIVE as a reader and a writer. If two processes are attempting this at the same time, the possibility of deadlock exists. Be sure to error out if you cannot lock the file, rather than wait for it to free up.

- Multiaccess : The main advantage to message files over other

means is the easy ability to access a file from many different jobs/sessions. Also, the file must be allowed to be accessed at least twice, once as a reader and also as a writer. Therefore an access mode of no multiaccess is changed to GMULTI automatically. The default is GMULTI, which means other job/sessions may access the file. MULTI means that only the current job/session may access the file (essentially it's the MAILBOX facility replaced).

FREAD

This reads a record of data from the begining of the queue. When a record is read, it is deleted and the next message comes to the front. Using FCONTROL 47 will prevent an FREAD from destroying the message. If a FREAD is performed against an empty file with no writers, an EOF is returned. If there are writers or if it is the first FREAD to the empty file, a wait will be in effect.

FWRITE

This writes a record of data to the end of the queue. If the file is full and there are no readers, an EOF condition will be returned. The FWRITE will wait on a full file if it is the first FWRITE after an FOPEN or an FCONTROL 45 is in effect. If the file is full and there are readers, the FWRITE will wait until a block of records has been read from the file.

FCLOSE

Closes the file. When a file is closed by a writer a close record is written to the file.

FCONTROL

There are a few additions to FCONTROL for IPC files.

control code

6 Write EOF. This will not actually write an EOF but it will write the file label and buffer area to disc; useful in helping the message file survive system crashes.

45 Extended wait. If set to TRUE readers will wait on an empty file that has no writers and writers will wait on a full file with no readers. If set to FALSE, an FREAD to an empty file with no writers, or an FWRITE to a full file with no readers, will return a EOF condition. The TRUE or FALSE will stay in effect until it is changed by another FCONTROL 45. Note: the process will also wait if it is the first FREAD or FWRITE after an FOPEN.

46 Writer's ID. If set to TRUE, each FREAD will cause the writer ID header to be placed in the first two words of the reader's target area. The first word and 2 for close record. The second word contains the ID. Open and close records contain only the writer ID. Data records contain the message after the header data. If set to FALSE, the open and close records are deleted immediately as they hit the head of the queue. The two-word header data is not passed on to the stack of the reader program.

47 Nondestructive read. If set to TRUE, the next FREAD will read

but not delete the message at the head of the queue. Subsequent FREADs will cause destructive reads unless another FCONTROL 47 is set. The next FREAD would also read the same message again. If set to FALSE, the next FREAD will cause the message to be deleted.

48 Interrupt procedure. This is used to pass the address of the interrupt handling routine. It stays in effect until another FCONTROL 48 is set. To disarm the trap, call with the parameter set to 0 (zero).

FCHECK

An additional message has been added to the catalog. It is error number 151 : CURRENT RECORD WAS LAST RECORD WRITTEN BEFORE SYSTEM CRASH. It just informs you that the last few messages written to the file may not have gone through before the system crashed. If you were expecting close records you probably will not be getting them.

FFILEINFO

With this intrinsic, the number of readers or writers and the address of the software interrupt procedure may be determined.

34 - returns the number of writers accessing the file

35 - returns the number of readers accessing the file

49 - returns the address of the software interrupt procedure

Software Interrupts on Message Files

Message files are the only files that allow for software interrupts. This feature allows for a process to sit in the background and use very little CPU.

Initially, your program is disabled for software interrupts. To enable, you must set the intrinsic FINTSTATE to TRUE. Setting it to FALSE disables software interrupts. Normally, before you enable FINTSTATE you want to set up the interrupt procedure. To arm the interrupt, use FCONTROL 48 and send the plabel address of the interrupt procedure. This will cause the file to be placed in no-wait I/O. Then initiate an FREAD on the file to be interrupted. When that is done you enable the interrupt by FINTSTATE(TRUE).

If you want to know the address of the plabel you have set in the FCONTROL 48, perform a FFILEINFO 49. This will return the plabel of the interrupt procedure currently set. If a 0 (zero) is return, no interrupt procedure is set.

When the file is interrupted your interrupt procedure is called, software interrupts are disabled. At this point your read is in the middle of the I/O. Because of the no-wait I/O mode of the file, you must issue an IODONTWAIT to complete the I/O. After you have performed your interrupt handling, you must exit with a FINTEXIT. This will re-enable software interrupts.

If a CONTROL-Y trap is also set in your program, it will disable software interrupts when it is tripped. When the CONTROL-Y interrupt is completed, the RESETCONTROL will restore the

software interrupt to it's previous value. If you want to be able to service a software interrupt in a CONTROL-Y trap, perform a FINTSTATE(TRUE) in your CONTROL-Y procedure.

There are several other things to note. There can be only one FREAD or FWRITE outstanding for any particular file. All additional FREADs or FWRITEs will be ignored. Software interrupts do not work on remote message files (files on other computers). Also, software interrupts will not interrupt while an MPE intrinsic or procedure is executing. The only exceptions are PAUSE, PAUSEX and IOWAIT. To abort an uncompleted FREAD or FWRITE, use FCONTROL 43 (abort nowait I/O).

Here is an example how a software interrupt would be set up.

```
Fnum := FOPEN (,,,);
IntAddress := @IntProcedure;
FCONTROL (Fnum,48,IntAddress);
If <> then  ErrorRtn (Fnum);
    .....
FREAD (Fnum,Target,Len);
If <> then  ErrorRtn (Fnum);
FINTSTATE (TRUE);
```

The interrupt procedure would be :

```
PROCEDURE IntProcedure;

Begin
  .....
  IODONTWAIT (Fnum,Command);
  If <> then  ErrorRtn (Fnum);
  CASE Command of
    Begin
      .....
    End;
  .....
  FINTEXIT;
End;    <<<< IntProcedure >>>>
```

Things to Watch For

There are little aspects to any facility that can cause a programer to look for hours at a particular problem. The following are observations and comments that might help you quickly find problems that you are experiencing using IPC message files.

If you are copying a message file using FCOPY, it will empty out your message file like any ordinary reader. But if you use the ;COPY option, the file will be opened in copy mode and will treat the file like a serial file. As a result it will completely copy the file without destroying the messages in the file. Note that to do this FCOPY has to have exclusive access. FCOPY without the COPY option is just like any other reader. As a result, if the file is empty upon opening, FCOPY will wait until a message is written. You will have to break and abort FCOPY or write a record to the message file.

You can open the file as a program in copy mode too (see FOPEN intrinsic details). This will give your program exclusive access. The file will be treated as a serial file. So you can read every message without destroying the previous one. This is the only mode where this will happen.

Remember that an EOF condition is a message that informs you that certain things have happened and does not neccessarily mean that the file is empty. As a reader, you will wait on an empty file if it is the first FREAD after an FOPEN or if there are writers associated with the file. Writers will wait on a full file if it is the first FWRITE after an FOPEN or if there are readers associated with the file. Readers will get an EOF

condition when the file is empty and the last writer closes the file. Writers will get a EOF condition if the file is full and the last reader closes the file.

If you are using software interrupts, remember that when the last writer leaves a file, an interrupt occurs. The interrupt will return an EOF. It's a way of telling you that all the writers have left. You either terminate the program, or if you want it to continue, close and then re-open the file.

One the features of IPC files is user mode no-wait I/O. But you can only get no-wait I/O when you initiate an FCONTROL 48 on the IPC file. So be sure to perform a FCONTROL 48 before you perform your first FREAD.

Uses for Message Files

As I have said earlier, if you want two or more processes to communicate, IPC message files are ideal. But then when would you want two processes to communicate. Well here are some examples.

Hewlett-Packard's HPMAIL product uses IPC message files. When you start up HPMAIL a background job, known as the mailroom, is streamed. This background job is responsible for the distribution of mail. It receives orders for mail distribution through an IPC message file.

Communication with background jobs is very useful. I have been working with one application where IPC has saved a port and added mobility. The application accepts data from HP3077a Datacap terminals and adds the data to a data base. At times we would like to adjust the time or see the reject rates or see if a datacap terminal is operational. To do this we have to be able to enter commands to the program. Initially we had a central terminal running the application program. This wasted a port since it was rare that we should enter commands to it. We wanted to place it as a job, thereby saving a terminal. But we still wanted to communicate with it to reset terminals, etc. Enter IPC message files. Using the software interrupt feature of IPC files, commands are entered to the application job and responses are returned via another IPC message file. This not only gave us an extra terminal, but also allowed us to communicate with the application from anywhere the system was accessible. When the application had a problem and needed reseting, we could do it from home via telephone modem.

Maybe all you want is a utility to send messages to the night operator. You could write a program that would add messages to an IPC message file. When the operator is ready to read them he runs another program that would print out all the messages. With MPE security you could arrange that only the operator can read it but anyone can write to it. This would save you from multiple editor files and keep messages secure from other mischevious people.

I am sure that there are many applications that I have not even approached. Just the ability to communicate across machines and with background jobs make message files extremely useful.

In Conclusion or So Why Use IPC Files

There are many uses for IPC files. But as always it depends on

how your application is to work. Don't use IPC files simply to use IPC files. But if you do find they will help your application you have many advantages going for you.

- uses the already known file intrinsics (FOPEN, FCLOSE, etc.)
- because of above, you can use high-level languages (BASIC would have to call SPL or FORTRAN procedures).
- utilizing file security system
- can use system file commands on file (BUILD, FILE, etc.)
- access remote IPC files via DS
- multiple user access
- user mode no-wait I/O on message files
- when writers leave, close records are written
- can identify which writer wrote a message
- software interrupts on message files (not directly available in COBOL)

The ease of use, complete error-checking and thought put into IPC communication makes IPC a very useful feature of the MPE file system.

References

1. Authors unknown, MPE Intrinsics Manual, December 1981 update, Hewlett-Packard Company.

2. ZEITMAN, Larry, "Software Interrupts", COMMUNICATOR, Issue 29, page 64-67, Hewlett-Packard Company.

FFILEINFO

Provides access to file information.

```
                    IV      IV      BA
FFILEINFO (filenum, [,itemnum1, itemvalue1]
                    [,itemnum2, itemvalue2]
                    [,itemnum3, itemvalue3]
                    [,itemnum4, itemvalue4]
                    [,itemnum5, itemvalue5] );
```

NOTE

Itemnum/itemvalue parameters must appear in pairs. Up to five items of information can be retrieved by specifying one or more *itemnum/itemvalue* pairs.

FFILEINFO provides access to file information. It is designed to be extensible so that new file information can be defined and accessed.

PARAMETERS

filenum — *integer by value (required)*
MPE file number returned by FOPEN

itemnum — *integer by value (optional)*
Cardinal number of the item desired; this specifies which item value is to be returned.

itemvalue — *byte array (optional)*
Value of the item specified by the corresponding itemnum; the data type of the item value depends on the item itself.

CONDITION CODES

CCE No error

CCG Not used

CCL Access or calling sequence error

ITEM #	TYPE	ITEM
33	I	label type
34	I	current number of writers
35	I	current number of readers
36	L	File Allocation Date (CALENDAR format)
37	D	File Allocation (CLOCK format)
38	L	SPOOFLE Device file number (#O or #I number)
39		RESERVED
40	D	disc or diskette device status
41	I	device type
42	I	device subtype
43	BA	environment file name
44	I	last disc extent allocated
45	BA	file name from labeled tape HDR1 record
46	I	tape density
47	I	DRT number
48	I	UNIT number
49	I	software interrupt PLABEL

FOPEN

INTRINSIC NUMBER 1

Used to establish access to a file and optionally define the physical characteristics of the file prior to setting up access to it.

```
                                  BA     LV     LV     IV     BA     BA
filenum:=FOPEN(formaldesignator,foptions,aoptions,recsize,device,formmsg,

                                  IV           IV           IV     DV           IV
                userlabel,blockfactor,numbuffers,filesize,numextents,

                                  IV     IV     0-V
                initialloc,filecode);
```

The FOPEN intrinsic makes it possible to access a file. In the FOPEN intrinsic call, a particular file may be referenced by its *formal file designator*, described in Section III. When the FOPEN intrinsic is executed, it returns to the user's process a *file number* by which the system uniquely identifies the file. This file number, rather than the file designator, then is used by subsequent intrinsics in referencing the file.

FUNCTIONAL RETURN

This intrinsic returns an integer file number used to identify the opened file in other intrinsic calls.

PARAMETERS

formaldesignator

byte array (optional)
Contains a string of ASCII characters interpreted as a formal file designator, as defined in Section III. This string must begin with a letter, contain alphanumeric characters, slashes, or periods, and terminate with any non-alphanumeric character except a slash or a period. If the string names a system-defined file, it can begin with a dollar sign ($); if it names a user-predefined file, it can begin with an asterisk (*). New KSAM files unlike standard files must be opened with a unique name.
Default: A temporary nameless file that can be read from or written to, but not saved, is assigned.

foptions

logical by value (optional)
The *foptions* parameter allows you to specify different file characteristics by setting corresponding bit groupings in a 16-bit word. The correspondence is from right to left, beginning with bit 15. These characteristics are as follows, proceeding from the rightmost bit groups to the leftmost bit groups in the word. The bit settings are summarized in figure 2-1.

NOTE

Bit groups are denoted using the standard SPL notation. Thus bits (14:2) indicates bits 14 and 15; bits (10:3) indicates bits 10, 11, and 12.

Bits (14:2) — Domain *Foption*.
The file domain to be searched by MPE to locate the file, indicated by these bit settings:
00 = The file is a *new* file, created at this point. No search is necessary.
01 = The file is an old permanent file, and the system file domain should be searched.
10 = The file is an old temporary file, and the job file domain should be searched.
11 = The file is an old file that is to be located by first searching the job file domain and then, if the file is not found, by searching the system file domain.

Bit (13:1) — ASCII/Binary *Foption*.
The code (ASCII or binary) in which a *new* file is to be recorded when it is written to a device that supports both codes. In the case of disc files, this also affects padding that can occur when a direct-write intrinsic call (FWRITEDIR) is issued to a record that lies beyond the current logical end-of-file indicator. In ASCII files, any dummy records between the previous end-of-file and the newly-written record are padded with blanks. In binary files, such records are padded with binary zeros. All files except mag tape and serial disc files are treated as ASCII files. Foreign disc files are binary records.
For ASCII files, this bit is 1.
For binary files, this bit is 0.

Bits (10:3) — Default File Designator *Foption*.
The *actual* file designator is equated with the formal file designator specified in FOPEN, if
1. No explicit or implicit :FILE command equating the formal file designator to a different actual file designator occurs in the job or session; *or*
2. The Disallow File Equation *Foption* (bit 5) is specified. (Note that a leading * in a formal designator can effectively override the disallow file equation *foption*.)

The bit settings are
000 = The actual file designator is the same as the formal file designator.
001 = The actual file designator is $STDLIST.
010 = The actual file designator is $NEWPASS.
011 = The actual file designator is $OLDPASS.
100 = The actual file designator is $STDIN.
101 = The actual file designator is $STDINX.
110 = The actual file designator is $NULL.

Bits (8:2) — Record Format *Foption*.
The format in which the records in the file are recorded, indicated by these bit settings:
00 = Fixed-length records. The file is composed of logical records of uniform length. Foreign discs always have fixed-length records.
01 = Variable-length records. The file contains logical records of varying length. This format is restricted to records that are written in sequential order. The size of each record is recorded internally. The actual physical record size used is determined by multiplying the *recsize* (specified or default) plus one by the *blockfactor*, and adding one word for the end-of-block indicator. This option is not allowed when NOBUF is specified. In such a case, the record format used is undefined-length records, discussed below.
10 = Undefined-length records. The file contains records of varying length that were not written using the variable-length *foption* (01). All files not on disc or magnetic tape are treated as containing undefined-length records by default. The file system makes no assumption about the amount of data that is useful. The user must determine how much data is good.
For undefined length records, only the data supplied is written with no information about its length.

Note: Undefined-length records are supported by all devices; fixed- and variable-length records are supported by disc and magnetic tape devices only. To state this another way: disc and magnetic tape devices support *all* record formats, whereas all other devices support *only* undefined length records.

Bit (7:1) — Carriage Control *Foption*.
If selected, this specifies that you will supply a carriage control directive in the calling sequence of each FWRITE call that writes records onto the file.
0 = No carriage control directive expected.
1 = Carriage control directive expected.
Carriage control is defined only for character oriented, i.e., ASCII, files. This option and binary are mutually exclusive and attempts to open new files with both binary and this option results in an access violation.

Bit (5:1) - Disallow File Equation *Foption*.
This option ignores any corresponding :FILE command, so that the specifications in the FOPEN call take effect (unless preempted by those in the file label, for disc files). Note that a leading * in a formal designator can effectively override the disallow file equation *foption*.
0 = Allow :FILE.
1 = Disallow :FILE.

Bits (2:3) File Type *Foption*
Determines the type of file to create for a new file. If the file is old, this field is ignored.

000 - Ordinary file
001 - KSAM file
010 - Relative I/O file
100 - Circular file (discussed in Section III)
110 - Message file

Note: The Default Designator FOPTION, bits 10 through 12, offers several choices for default file designators. Any value used other than 0 for "filename" will override the File Type field.

Bits (0:2) — Reserved for MPE. Should be set to zero.

aoptions *logical by value (optional)*

The *aoptions* parameter permits you to specify up to seven different access options established by bit groupings in a 16-bit word. These access options are described below. The bit settings are summarized in figure 2-2.

Bits (12:4) — Access Type *Aoptions*.

The type of access allowed for this access of this file:

0000 = Read access only. The FWRITE, FUPDATE, and FWRITEDIR intrinsic calls cannot reference this file. The end-of-file is not changed, the record pointer starts at 0.

0001 = Write access only. Any data written in the file prior to the current FOPEN request is deleted. The FREAD, FREADSEEK, FUPDATE, and FREADDIR intrinsic calls cannot reference this file. The end-of-file is set to 0; the record pointer starts at 0. On magnetic tape an EOF mark will be written to the tape when the file is FCLOSED even if no data is written.

0010 = Write access only, but previous data in the file is not deleted. The FREAD, FREADSEEK, FUPDATE, and FREADDIR intrinsic calls cannot reference this file. The end-of-file pointer is not changed, the record pointer starts at 0. Therefore, data will be overwritten if a WRITE is done.

0011 = Append access only. The FREAD, FREADDIR, FREADSEEK, FUPDATE, FSPACE, FPOINT, and FWRITEDIR intrinsic calls cannot reference this file. This option is not valid for files containing variable-length records. The end-of-file pointer is used to set the record pointer prior to each FWRITE. For disc files it is updated (in an internal file system table) after each FWRITE. Thus, data in the file cannot be overwritten.

0100 = Input/output access. Any file intrinsic except FUPDATE can be issued for this file. The end-of-file pointer is not changed, the record pointer starts at 0.

0101 = Update access. All file intrinsics, including FUPDATE, can be issued for file. The end-of-file pointer is not changed; the record pointer starts at 0.

0110 = Execute access. Allows user with Privileged Mode Capability input/output access to any loaded file. The end-of-file pointer is not changed, the record pointer starts at 0.

Bits (8:2) — Exclusive *Aoption*.

This *aoption* specifies whether you have continuous exclusive access to this file, from the time it is opened to the time it is closed. This option often is used when performing some critical operation, such as updating the file.

01 = Exclusive access. After this file is opened, prohibits another FOPEN request, whether issued by this or another process, until this process issued the FCLOSE request or terminates. If any process already is accessing this file when this FOPEN call is issued, a CCL error code is returned to the calling process. If another FOPEN call is issued for this file while the exclusive *aoption* is in effect, an error code is returned to that calling process. The exclusive access *aoption* can be requested only by users allowed the file locking access mode by the security provisions for the file.

10 = Semi-exclusive access. After the file is opened, prohibits concurrent output access to this file through another FOPEN request, whether issued by this or another process, until this process issues the FCLOSE request or terminates. A subsequent request for the input/output or update *aoption* access type will obtain read only access. Other types of read access, however, are allowed. If any process already has output access to the file when this FOPEN call is issued, a CCL error code is returned to the calling process. If another FOPEN call that violates the read-only restriction is issued while the semi-exclusive *aoption* is in effect, that call fails and an error code is returned to the calling process. The semi-exclusive access can be requested only by users allowed the file-locking access mode by the security provisions for the file.

11 = Share access. After the file is opened, permits concurrent access to this file by any process, in any access mode, subject to other basic MPE security provisions in effect.

00 = Default value. If the read access only *aoption* is selected, share access (11) takes effect. Otherwise, exclusive access (01) takes effect. Regardless of which access is selected, FGETINFO will report 00.

Bit (7:1) — Inhibit Buffering *Aoption*.

When selected, this *aoption* inhibits automatic buffering by MPE and allows input/output to take place directly between the user's data area and the applicable hardware device.

0 = Allow normal buffering.

1 = Inhibit buffering (NOBUF).

NOBUF access is oriented to the transfer of physical blocks rather than logical records.

With NOBUF access, you have responsibility for blocking and deblocking of records in the file (see Section III). To be consistent with files built using buffered I/O, records should begin on word boundaries, and when the information content of the record is less than the defined record length, the record should be padded with blanks by you if the file is ASCII or with zeros if the file is binary.

The *recsize* and block size for files manipulated under NOBUF access follow the same rules as those files that are created using buffering. The default *blockfactor* for a file created under NOBUF is one.

When a NOBUF file is opened without multirecord access, the amount of data transferred per read or write is limited to a maximum of one block.

The end-of-file, next record pointer, and record transfer count are maintained in terms of logical records for all files. The number of logical records affected by each transfer is determined from the size of the transfer.

Transfers always begin on a block boundary. Those transfers which do not transfer whole blocks leave the next record pointer set to the first record in the next block. The end-of-file pointer always points at the last record in the file.

For files opened with NOBUF access, the FREADDIR, FWRITEDIR, and FPOINT intrinsics treat the *recnum* parameter as a block number.

Non-RIO access to a RIO file can be indicated by specifying the NOBUF option. In this case the physical blocksize (item #21) from FFILEINFO should be used to determine the maximum transfer length. (The FGETINFO "blksize" parameter may also be used.)

Bits (5:2) — Multi-Access Mode *Aoption*

This feature permits processes located in different jobs or sessions to open the same file.

00 – No multi-access.

01 – Only intra-job multi-access allowed; this is the same as specifying the MULTI option in a FILE command.

10 – Inter-job multi-access allowed; this is the same as specifying the GMULTI option in a FILE command.

11 – Undefined. If this is specified, the FOPEN will be rejected with an error code of 40: ACCESS VIOLATION.

Bit (4:1) — No-Wait I/O *Aoption*.

The selection of this *aoption* allows you to initiate an I/O request and to have control returned before the completion of the I/O. The IOWAIT intrinsic must be called after each I/O request to confirm the completion of the I/O. The No-Wait I/O *aoption* implies the NOBUF *aoption*; if you do not specify NOBUF, the file system does it for you. Also, multirecord access is not available. This option is not available if the file is located on a remote computer.

NOTE

You must be running in Privileged Mode to use No-Wait I/O and NOBUF.

Bits (3:1) — File Copy *Aoption*

This feature permits any file to be treated as a standard sequential file, rather than as a file of its own type.

0 – The file will be accessed in its native mode; that is, a message file will be treated as a message file, a KSAM file as a KSAM file, etc.

1 – The file is to be treated as a standard, sequential file with variable-length records.

Note: In order to access a message file in copy mode, a process must have exclusive access to the file.

Bits (0:3) — Reserved for MPE. Should be set to zero.

Default: All bits are set to zero.

CONDITION CODES

CCE Request granted. The file is open.

CCG Not returned by this intrinsic.

CCL Request denied. This may be because another process already has exclusive or semi-exclusive access for this file, or an initial allocation of disc space cannot be made due to lack of disc space. The file number value returned by FOPEN if the file is not opened successfully is zero. The FCHECK intrinsic should be called for more details.

IODONTWAIT Initiates completion operations for an I/O request.

```
                I                       IV   LA     I       L  O-V
fnum:=IODONTWAIT(filenum,target,tcount,cstation);
```

If IOWAIT is called and no I/O has completed, then the calling process is suspended until some I/O completes; if IODONTWAIT is called and no I/O has completed, then control is returned to the calling process (CCE is returned and the result of IODONTWAIT is zero.)

FUNCTIONAL RETURN

This intrinsic returns an integer representing the file number for which the completion occurred. If no completion occurred, zero is returned.

PARAMETERS

filenum — *integer by value (required)*
A word identifier specifying the file number for which there is a pending I/O request. If zero is specified, the IODONTWAIT intrinsic will check for any I/O completion.

target — *logical array (optional)*
A word pointer specifying the DB-relative address of the user's input buffer. This buffer must large enough to contain the input record. It should be the same buffer specified in the original I/O request if that request was a read. This allows for proper recognition of :EOD (AND :) where applicable.

tcount — *integer (optional)*
A word to which is returned a positive integer representing the length of the received or transmitted record. If the original request specified a byte count, the integer represents bytes; if the request specified words, the integer represents words. Note that this parameter is pertinent only if the original request was a read. The FREAD intrinsic always returns zero as its functional return if no-wait I/O is specified. In this case, the actual record length is returned in the *tcount* parameter of IODONTWAIT.
Default: The length of the record is not returned.

cstation — *logical (optional)*
Used for distributed systems to return the number of the calling station which completed.

CONDITION CODES

CCE	Request granted. If the functional return is non-zero then I/O completion occurred with no errors. If the return is zero then no I/O has completed.
CCG	An end-of-file condition was encountered.
CCL	Request denied. Normal I/O completion did not occur because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

SPECIAL CONSIDERATIONS

You must be running in Privileged Mode to specify FOPEN *aoptions* No-Wait I/O.

With IPC, FCONTROL 48 causes No-Wait I/O in user mode.

FCONTROL Performs control operations on a file or device.

```
                        IV      IV     L
FCONTROL(filenum,controlcode,param);
```

The FCONTROL intrinsic performs various control operations on a file or on the device on which the file resides.

PARAMETERS

filenum — *integer by value (required)*
A word identifier supplying the file number of the file for which the control operation is to be performed.

controlcode — *integer by value (required)*
An integer specifying the operation to be performed:

4 - Set Time-Out Interval. This code indicates that a time-out interval is to be applied to input from the terminal. If input is requested from the terminal but is not received in this interval, the FREAD request terminates prematurely with condition code CCL. The interval itself is specified, in seconds, in a word on the user's stack, indicated by *param*. If this interval is zero, any previously established interval is cancelled, and no time out occurs. Controlcode 4 is ignored if the addressed file is not being read from the terminal. Note that this only affects the next read.

43 = Aborts pending NO-WAIT I/O request.

45 = Enable/Disable extended wait.

46 = Enable/Disable reading writer's ID.

47 = Nondestructive read.

48= Set software interrupt procedure trap

param — *logical (required)*

If *controlcode* is 2, 5, 6, 7, 8, or 9, *param* is any variable or word identifier. This parameter is needed by FCONTROL to satisfy the internal requirements of the intrinsic. It serves no other purpose, however, and is not modified by the intrinsic.

for 48 give address of interrupt procedure

See Section V for *param* requirements when *controlcode* is 10 or greater.

CONDITION CODES

CCE	Request granted.
CCG	Not returned by this intrinsic.
CCL	Request denied because an error occurred.

SPECIAL CONSIDERATIONS

Split stack calls permitted.

SETTING UP SHOP - GETTING AND KEEPING ON THE RIGHT TRACK

Linda Pottenger
Hewlett-Packard

You have just purchased a Hewlett-Packard 3000 system, and have aquired the "friendly" hardware/software that comes as standard with your 3000. You have a staff that is eager to produce good systems for your users. With these raw materials you will attempt to create a friendly atmosphere for yourselves, as data processing professionals, and for your users. So, do yourself a favor and set up shop right. Make the lives of both yourselves and your users an easy, efficient one by establishing some basic ground rules and sticking to them.

Establishing standards and procedures will take some time on your part, however this investment will buy you several things in the long run.

1. New accounts/systems can be set up easily because the wheel has already been invented. The thought-time to create a new process is reduced because it becomes a "routine" rather than a "task".

2. The education curve will be lower for your new hires (both data processing and users). A published standards and procedures manual can be used as a reference for training new employees as well as a refresher for "old-hands".

3. Maintenance activities can be shortened because systems/programs can be identified and tested easily. Users will find it easier to converse with data processing personnel when they can talk in terms of a documented program number that has a bug, rather than the "gross pay report that I run every other Tuesday doesn't work".

4. Being organized conveys a subtle message to your users that you know what you are doing. You appear much more competitent when you can immediately locate a file rather than when you must search the entire system to locate that same file. Your user's time is just as valuable to him, as yours is to you. Being able to respond quickly is one of many ways to increase your credibility.

Let's now investigate a few ways to help you set up shop.

1. ACCOUNT STRUCTURE MODEL

Create a standard model to be used for EVERY account. Customize only when necessary. Some of the benefits of this standard structure is that you will always find data bases in the SAME group, source code in the SAME group, the account manager is ALWAYS THE SAME user name, and the groups and accounts are secured equally throughout the accounts.

SAMPLE ACCOUNT STRUCTURE

ACCOUNT NAME: _____ ACCOUNT PASSWORD: _____

ACCOUNT DESCRIPTION: _____

GROUP NAME:	FILE	JOB	JOBC	PUB	SOURCE
GROUP PASS: CONTENTS: SECURITY:					
USER NAME: USER PASS: PURPOSE:					
USER NAME: USER PASS: PURPOSE:					
USER NAME: USER PASS: PURPOSE:					

DISTRIBUTION: _____

_____

_____

System manager

2. NAMING CONVENTIONS

Naming files in a conventional manner allows immediate identification of the file. Set up standards for file names and STICK TO THEM. You can't be very descriptive with 8 characters, so develop a naming convention that can become descriptive to you.

Standardize these names as a minimum requirement:
a. Application programs
b. Job streams
c. USL files
d. UDC files
e. Permanent files
f. VPLUS files
g. KSAM files
h. Data bases
i. Report names

Suggested naming convention:

Create an 8 character file name formatted as follows:

TAAFFNNN

where:

T = Type of file
AA = Application
FF = Function
NNN= Sequential number identifier

a. APPLICATION PROGRAMS

T = type of file
"S" = source code
"X" = executable code

AA = application
"AR" = accounts payable
"PY" = payroll, etc.

FF = function
"EX" = report extract program
"MN" = main menu or driver program
"RP" = report writer
"UD" = update
"UT" = utility

NNN = sequential number identifier

All programs that are "COMPANION" programs will share the same sequential number identifier. For example, a companion extract and print program will be named: XAPAR100 and XAPRP100.

b. JOB STREAMS

T = type of file
"J" = application job stream
"C" = compile job stream

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
Same as APPLICATION PROGRAM definition.

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition.

All job streams will carry the same name as the main program it executes.

c. USL FILES

T = type of file
"U" = user file

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
"SL" for segmented library

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition. This number is the same as the executable code module that it creates.
an example: SPYUD100, SPYUD101, SPYUD102 are compiled into UPYSL100 to create XPYUD100

d. UDC FILES

T = type of file
"U" = user file

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
"DC" for defined commands

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition.

e. PERMANENT FILES

T = type of file
"W" = work file

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
"S1" for sequential file #1, "S2" for #2, etc.

NNN = sequential program identifier
Same as application program definition. This number is the same as the executable code module that creates it.

f. VPLUS FILES

T = type
"V" = VPLUS form file
"F" = VPLUS fast form file

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
Same as APPLICATION PROGRAM definition.

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition.

g. KSAM FILES

T = type
"K" = KSAM

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
"KY" = KSAM KEY FILE
"DT" = KSAM DATA FILE

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition.

h. DATA BASES

Since IMAGE allows 6 characters for a data base name, a deviation from the above naming convention is required.

TT = type
"DB" = data base

AAAA = application
Same as APPLICATION PROGRAM definition, however may have 2 more characters for clarity.

The data base schema should be named in the above manner, with "SC" as the last two characters of the 8 character name.

i. REPORT NAMES

T = type
"R" = report

AA = application
Same as APPLICATION PROGRAM definition.

FF = function
Same as APPLICATION PROGRAM definition.

NNN = sequential program identifier
Same as APPLICATION PROGRAM definition. This numbering convention allows for immediate recognition of program number when the report number is known.

3. DOCUMENTATION STANDARDS

Documentation, though an undesirable task, is a necessity to every data processing shop. This documentation does not need to be elaborate. It needs to be consise, and useful.

Some programming languages lend themselves to comments. COBOL for example, can be self documenting if certain program standards for documentation are enforced. Require COBOL programs to contain a purpose, a description, and a brief history of program changes in the remarks section. Require that sections or paragraphs be commented if the section or paragraph performs complex routines or calculations. Require COPYLIB buffers, data names, and standard routines be used.

Require on paper a few minimums:

a. Cross-reference of program number to program description.
b. Cross-reference of program number to job stream reference.
c. Cross-reference of program number to files used.
d. Data dictionary
e. Report definition

4. SERVICE REQUESTS

In order to be able to prioritize requests of your time, some mechanism must be provided to respond to service requests. A suggested way to do this is to establish a SERVICE REQUEST system. This may either be a manual or mechanized system. The minimum requirement is to develop a form to be completed if data processing services are required. This form becomes your authorization to make changes to existing systems, or to begin development on new systems. This form provides you with documentation as to why a change was made, when the change was made, and at who's request the change was made. It becomes a great historical document for your systems development history.

To aid you in prioritizing your time, these requests can have estimated completion time vs benefits gained documented on the form. This is an aid to your steering committee in determining who gets what done first.

As projects are completed, completed forms can be filed in a "COMPLETED" binder. Outstanding projects can be filed in a "PENDING" binder. If you include such things as date requested, department requesting, required completion date, time required, then you can prioritize in a manner most meaningful to you.

A sample SERVICE REQUEST form follows.

REQUEST FOR SERVICES REQUEST NO. _____
YYMMNNNN

DESCRIPTION OF SERVICES REQUESTED: _____

SAMPLE OF REQUEST OR PROBLEM ATTACHED?

REQUESTED COMPLETION DATE	REQUEST INITIATED BY	REQUEST APPROVED BY

☐ SERIOUS PROBLEM ☐ MINOR PROBLEM ☐ ENHANCENCEMENT ☐ INFORMATION ONLY

DATA PROCESSING USE ONLY

DATE RECEIVED	REQUEST NAME	APPLICATION	DEPARTMENT	PRIORITY	ASSIGNED TO

RESOURCE SUMMARY	ESTIMATED TIME	ACTUAL TIME	DIFFERENCE
MAN-HOURS			
CALENDAR DAYS			
START DATE			
COMPLETION DATE			
OTHER RESOURCES (OPERATIONS HARDWARE SUPPORT)			

ESTIMATED PROJECTION COMPLETED BY/DATE	ACTUAL TASK COMPLETED BY/DATE

MANAGEMENT APPROVALS	USER	DATA PROCESSING
REQUEST APPROVED		
FINAL ACCEPTANCE		

COMMENTS:

5. PROGRAM STANDARDS

In addition to following the naming conventions as sited earlier, a few more program standards should be followed.

a. Document internally in your program. Always state the purpose, run time considerations, program change history. Always comment any difficult routines or calculations.

b. Label your routines so that it is easy to find a particular paragraph. A numeric prefix to a paragraph is always a good idea.

c. Code one statement per line. It is easier for the eye to follow logic if it is pleasing to the eye, and easy to digest with a columnar scan.

d. Make data names meaningful. Gross-pay is a more descriptive name than G1.

e. Use indentation when coding conditional logic. It is easier to follow the if/then construct if the statements are aligned in the code.

f. Keep communication to the console at a minimum. Too many messages begin to be ignored by the operator if they are not critical. It then becomes more difficult to sort out the critical messages from the "chatty" ones.

g. Use standard COPYLIB buffer descriptions, and standard routines. This insures continuity of data names and routines throughout a system. Develop standard error handling and use it consistantly.

h. Develop a standard heading for all reports. Include report number, run date, run time, page number, and descriptive title.

i. Develop a standard heading for all screens. Include screen number, program number.

j. Edit all input closest to the source to detect errors.

k. Establish a standard means to enter date fields (MMDDYY), as well as storage of date fields (YYMMDD).

l. Require testing of all conditions in a program. These should include validity checks, file updates, volume tests, exception processing, logic testing, calculation routines, as well as checking any special processing considerations such as month or year-end processing, or control totaling.

6. JOB STREAM STANDARDS

In addition to the naming conventions previously mentioned, consider the following for job stream standards.

a. Build files needed for work files only in the job stream needed. Purge the file after usage.

b. Comment the job stream's usage and estimated run schedule.

c. Comment any special run time considerations needed for this job. If this job is dependent on sucessful completion of another job, allow the streaming of the job to be spawned from the successful run.

d. Use of JCW's is helpful if conditional steps to be executed in the same stream.

e. Limit TELLOP messages unless a critical condition occurs.

f. Use the FORMS option on files where print is defered to tell the operator why the print is deferred.

Develop your own set of standards and procedures. USE them, document them, distribute them. Let your staff as well as your users know that these are the rules you are playing by.

In conclusion, use of any or all of the above observations should be helpful to you in setting up shop. Whether you've been a user 2 weeks or 2 years, organization is key in keeping on track.

DATA DICTIONARIES -- A NEW ERA

Gary Puckering
Quasar Systems Ltd.

Ten years ago, the Data Dictionary concept began to emerge. Although not given serious attention within the DP community at first, by the late 1970's many companies were making use of the new Data Base Management systems and Inquiry/Report languages to turn over many DP tasks to the end-user. To do this effectively, however, required that the end-user know what data was available and in what form. Unfortuately, most of this information was buried inside the program code as data definitions, in a format understandable only to those who spoke the language -- the programmmers!

In response to this, many organizations undertook the task of cataloging their data with the idea of producing a master catalogue of data definitions for end-user reference. The approach to this task was influenced greatly by the concepts of data integration and data independence, concepts which were being strongly promoted by the Data Base Management systems. Data elements were examined in a different light, independent of the files and programs that contained or manipulated them. The realization of Data as a Corporate Resource had finally emerged.

Following this realization, many companies acquired or developed an automated data dictionary system as a tool to manage this newly-discovered resource. Soon, the data dictionary became the central repository of data definitions in many installations. New software tools emerged to extract data definitions from the dictionary, a process which began to replace the function of the traditional Copy libraries.

The next major advance in software development, dictionary-based programming languages, was not long in coming. Within these new languages, the separation of data definition and processing specification was complete. Now, the application programs described only the processes to take place on the data. The data itself was described in the data dictionary.

The potential for further advances in software over the next few years lies largely in new possibilities opened up by dictionary-based programming languages. The purpose of this paper is to describe some of these possibilities, as well as to define many of the important problems that must be overcome in order to provide practical and useful dictionary systems for use by dictionary-based programming languages.

Terminology

The following terms will be used to describe the contents of data dictionaries in subsequent discussions:

1) Object-type — An object-type is a class of dictionary objects which represent real-world facts, concepts or instructions. For example, FILE, ELEMENT, PROGRAM, or LOCATION.

2) Object — An object is a specific occurrence of an object-type. For example, FILE VENDOR, ELEMENT VENDOR-NAME, PROGRAM VENDOR-LIST, or LOCATION PURCH.MMS.

3) Attribute — An attribute of an object-type is a named characteristic of that object-type. For example, CAPACITY is an attribute of FILE, COLUMN-HEADING is an attribute of ELEMENT, DATE-COMPILED is an attribute of PROGRAM.

4) Attribute-Value — An attribute value is the specific value of an attribute associated with an object-type for a given object. For example, the value of CAPACITY for FILE VENDOR is 10000, the value of COLUMN-HEADING for ELEMENT VENDOR-NAME is "Vendor Name", etc.

5) Relations — A relation is a way of describing a connection or association between two or more object-types. For example, FILE CONTAINS ELEMENT specifies an association between the object-types FILE and ELEMENT.

6) Relationships A relationship is a specific occurrence of a relation. For example, FILE VENDOR CONTAINS ELEMENT VENDOR-NAME.

Types of Data Dictionary Systems

Generally speaking, there are two types of dictionary systems: batch-oriented and interactive. Batch-oriented dictionary systems are similar to language compilers. Basically, they accept a data definition language, usually similar in appearance to the COBOL Data Division, parse it, and then create (or update) the actual data dictionary. Interactive dictionary systems, on the other hand, usually update the data dictionary directly. They are on-line systems which interact with the user either by prompt-and-response or through menu and data screens.

Both types of dictionary systems have their advantages. For example, in a batch-oriented dictionary system, a series of changes can be made to the source code and when it is time to put these changes into effect, they can be compiled into the dictionary all at once.

Interactive dictionary systems avoid the overhead of compilation by updating the dictionary directly. This usually results in improved productivity during application development, since the dictionary can be changed more quickly, but can present problems during production in coordinating dictionary changes with program changes.

Some dictionary systems are actually a hybrid of the two types just discussed. These systems allow the user to maintain the source code in many small units, such as a record layout. Each unit must be compiled into the dictionary but making changes is usually faster than in completely batch-oriented dictionary systems since only the affected units of source code need be recompiled.

Because interactive data dictionaries are more responsive to changes, the trend in dictionary systems is in this direction. However, as new dictionary systems are developed over the coming years, much more attention to the problem of controlling changes will have to be given.

Version control

One method of controlling changes in a data dictionary is through version control. With this approach, each object in the dictionary can exist as one or more versions. Each version can have different attribute-values or

relationships.

For example, the following definitions might exist in a dictionary:

Element	Version	Type	Allowed-Values
REGION-CD	0	9(2)	01 to 09
REGION-CD	1	9(2)	01 to 10
REGION-CD	2	9(2)	01 to 10, 15
VENDOR-NAME	0	X(30)	

In this example, three different versions of REGION-CD exist, each with a different Allowed-Values attribute, but only one version of VENDOR-NAME exists.

To make effective use of this facility, the programming languages which reference these dictionary data definitions must be able to select a version based upon some user-supplied criterion, such as a Version Limit. For example, a version limit of 1 would cause VENDOR-NAME version 0 and REGION-CD version 1 to be selected, since these are the highest versions present within the version limit.

To fully address the problem of version control, future application languages must provide a similar facility for coordinating program versions with dictionary versions. If, for example, object modules were kept track of in the dictionary, then the version limit that was in effect at the time of compilation could be carried as an attribute. Similarly, source modules could be kept track of in the dictionary (also with a version attribute).

With this approach, it would be possible to carry out development and production using the same dictionary. Changes to an existing system could be made by introducing new versions of data definitions or programs without affecting the current production system. When the changes have been tested, the new versions of programs and data definitions can be promoted to the production-level version number with a utility program.

Active Updating

The prospect of keeping track of programs in the dictionary points to yet another feature of dictionary-based software which will emerge over the next few years: Active Updating. As previously mentioned, current dictionary-based language compilers (or translators) obtain data definitions for their programs by directly accessing the dictionary. However, the flow of

information is one-way only. Since it is equally important to keep track of programs as it is to keep track of data definitions, making communication between the program compilers and the dictionary a two-way street is an important next step in the evolution of dictionary-based software.

To support this facility, new dictionary systems will have to provide object-types for various process entities. For example:

Object-type	Attribute
REPORT-PROGRAM	REPORT-NAME
	REPORT-DEVICE
	REPORT-FORMS
	REPORT-COPIES
	REPORT-PRIORITY
	REPORT-SPACING
	PAGE-TITLE
	PAGE-LENGTH
	PAGE-WIDTH
	PAGE-IMAGES
ONLINE-SCREEN	SCREEN-TYPE (Menu/Data)
	ACTIVITIES-ALLOWED
	START-LINE
	SCREEN-LENGTH
	WINDOW-LINE
	MESSAGE-LINE
BATCH-PROCESS	COMPILED-BY
	DATE-COMPILED
	PROCESS-LIMIT
	REPORT-DEVICE
	LOCK-OPTION

Once these object-types and attributes were defined in a dictionary, then objects and attribute-values could be automatically generated whenever a program is compiled. This would provide an accurate, up-to-date source of information on processes which reference the dictionary.

Perhaps of even greater value, though, is the potential of having compilers automatically generate relationships between data and process objects. Such a feature can provide a wealth of information which is invaluable for ongoing maintenance and enhancement of systems.

For example, whenever a program is compiled, the relationship between it and the files and elements it uses can be automatically recorded in the dictionary. Moreover, information about the way in which these data objects are used (such as whether the file is read,

written or updated, or whether a particular element is modified, displayed, used in calculations, or not used at all) can also recorded. Such information can be used to determine which programs may be affected by a change to a data definition.

Initially, dictionary systems will be able to provide Impact Assessment reports which simply identify the programs which may be affected by a given change. Ultimately, however, these systems will be able to differentiate between changes that necessitate a recompilation and changes that require a redesign. For instance, changing the allowed values of an element may necessitate a recompilation of the programs which use that element, but increasing the size of an element may require that screens or reports be redesigned to accomodate a larger field.

In addition to reporting such information, future dictionary systems will help reduce the maintenance effort by generating the job control commands necessary to recompile affected programs.

User Extensibility

User extensibility is a feature of dictionary systems which allows the dictionary user to define his own object-types, attributes and relations in the dictionary. With it, he can extend the standard definitions provided in the dictionary to tailor it to his own environment.

Many dictionary systems already support user extensibility, to the extent that standard commands or screens are provided to enter and maintain user-defined objects, attribute-values and relationships. In general, however, the users ability to create customized screens, commands or reports, or provide special edit checks, is quite limited since this requires programmatic access to the dictionary.

Once again, the potential for improvement here lies in improving the interaction between fourth-generation languages and the dictionary. New access methods can be introduced which will allow the user to develop programs which maintain and report user-defined objects in the dictionary.

For example, the user could define a new object-type, such as FORM, with attributes such as FORM-NO, FORM-NAME, FORM-TYPE, SIGNED-BY, etc. He could then write a program to enter and maintain forms in the dictionary by declaring FORM as a dictionary file. Data which is read from this "file" is actually read from the dictionary. A form

"record" would consist of the elements FORM-NO, FORM-NAME, FORM-TYPE, etc. This gives the dictionary user a high-level programmatic interface to the dictionary.

Programming Language Interface

Despite the tremendous growth in popularity of fourth-generation languages, conventional languages like COBOL and PL/1 will remain in use for many years to come. Therefore, dictionary system designers must provide interface facilities for these languages.

To support conventional languages, dictionary systems must provide utilities to generate COBOL FD's, PL/1 Declare statements, etc. In addition, interfaces to various data base management systems can be provided in a similar way by generating the appropriate DBMS data definition source code. Once again, the dictionary is to be treated as the central repository of data definition information.

Another interface method that is gaining increasing importance as dictionary systems become more widespread is a standardized programmatic interface; that is, a standard set of GET/PUT routines which allow programs to retrieve and store data in the dictionary. Such routines make possible two things. First, it makes possible the development of custom programs which maintain dictionary data or report dictionary contents. Second, it permits development of routines which provide special functions like editing or output formatting by referencing dictionary information.

Conclusion

Data dictionaries, and dictionary systems, have come a long way in the last few years. Already they have had a significant impact on the design of application-development tools. The trend toward dictionary-based programming languages is bound to accelerate over the next few years as more and more software developers respond to the demand for them and as the technology of data dictionary systems continues to mature.

IS THE FOURTH-GENERATION SOFTWARE LIVING UP TO ITS EXPECTATIONS?

William A. Rose, III
Noesis Computing Company

More and more fourth-generation software products are becoming available for the HP/3000, and the advertised advantages of these products seem impressive. The developers of fourth-generation software claim that users will save time and money on software systems development. It is therefore important that the usefulness of fourth-generation software products be evaluated for those people interested in the advantages of this type of software and also for those people who may already be using the products without receiving the full benefit from their investments. In this paper I intend to explore the use of fourth-generation software on the HP/3000 and suggest some general principles for maximizing its usefulness.

Unless you are involved in the study of different types of programming languages, you may be asking yourself the question "What are fourth-generation software products?" The label "fourth-generation" software is used to describe a relatively new and fast-growing group of products. RAPID/3000 from Hewlett-Packard and the Powerhouse products from Quasar Systems Ltd. are two examples that are available for the HP/3000.

James Martin, a well-known author on computer software, stated his definition of fourth-generation software in his book Application Development Without Programmers. I would like to use his definition as a guideline in discussing and evaluating this new type of software. Martin states two criteria that should be met in order to classify a language as fourth-generation. The first is that a non-technical, non-programmer, end user should be able to take a two-day course and obtain a working knowledge of the language. The second criterion is that a fourth generation software should enable software applications to be produced in one-tenth the time that it would take using conventional programming languages (e.g. COBOL, PASCAL

or FORTRAN). (If you are wondering about the first three generations, they are machine code, assembly language, and higher-level languages such as COBOL, PASCAL and FORTRAN that are machine independent.)

Before proceeding, I would like to make a few comments. The terms "fourth-generation software products" and "fourth-generation languages" have been used interchangeably, although the latter term is more descriptive. However, the developers of these products have tended to avoid using the term "language" and I will try to do the same. The experience that I will draw from for this discussion is with the Powerhouse products from Quasar Systems Ltd. But the areas that will be explored are general enough to apply to other fourth-generation software products.

Also, in an effort to simplify this discussion, I would like to categorize the products into three general areas: database description languages, screen-generators and report-writers. Since fourth-generation report-writers are the most widely used of the three, the emphasis will be in this area. This discussion can be expanded into other areas, but please keep in mind that you will probably be leaving the realm of end users.

Martin's first criterion for a fourth-generation language is that an end user should be able to gain a working knowledge of the language from a two-day course. Important implications for software productivity flow from this criterion. But before these implications are described, it should first be established that Martin's criterion can be met. Have end users been able to learn and effectively use fourth-generation software products?

There is no doubt that the answer to this question is "yes". Any vendor of fourth-generation software can and will gladly place you in contact with many such people who have had good experiences in solving their data processing problems in a cost-effective manner. However, there are other users who cannot tell such a happy story. If you are one of these frustrated people or are thinking of taking a fourth-generation language course, the following guidelines may help you avoid wasting your time and money.

The most basic guideline is also the most important: make sure that you or the person that you are sending to the class really wants to learn the material. When an organization or a department is first exposed to fourth-generation software, there is usually a likely person who ends up with the role of "pioneer". There is often a lot of pressure on the pioneer to be bold and lead the way. Unfortunately, if this person is not really interested but only yielding to pressure, he or she is likely to end up frustrated with

fourth-generation software. It's not in anyone's best interest to put a reluctant person on the spot.

Before you attend a class, check to see if hands-on terminal experience will be allowed. If you are going to be expected to sit down at a terminal and write a program, familiarize yourself beforehand with the editor that you will use during the class. Trying to write a program in a new language using a new writing tool can be a very frustrating experience. In fact, it is analogous to placing someone on a horse and expecting him to race in the Kentucky Derby before you teach him how to ride. Try to learn the basics of the editor before the class if possible.

Another unfortunate situation happens all too often. Someone takes a class and then returns to a busy business environment. The person's workload makes it difficult to take time and practice using the knowledge gained from the class. Several weeks or months later when the workload lightens up and the person starts using the software, the class material seems foreign. It is important to allocate some time immediately after a class to practice using what you just learned.

A guideline closely related to the previous one has to do with familiarity. Usually people will be attending a class in order to learn how to write reports from an existing database that is important to their work, but the examples used in a classroom situation revolve around a generalized database. It is helpful to become familiar with your actual database before you take a class. You will be able to ask questions that relate to your real-life situation and apply the material in a way that is beneficial to you. Try to be familiar with your database so that a class will be more meaningful and you will have a context in which to practice after the class.

The last guideline that I will propose concerning the idea of a two-day class relates to the course material. If classes are taught at different levels, make sure that you take the correct one for you. If you don't have any options and you find your course proceeding too quickly, let the instructor know that you are having difficulties. Non-technical users can swiftly lose interest in a class full of programmers.

The conclusion that I draw concerning Martin's first criterion is "yes", there does exist software that can be learned by end users in a two-day class. Care should be exercised in selecting people to attend a class, preparing for the class and allocating follow-up time afterward. But the ability to transform end users into people who can satisfy most of their own data processing needs with little technical support is a very powerful argument for the usefulness of fourth-generation software products.

Turning to a more technical topic, I would like to discuss Martin's second criterion for fourth-generation languages: applications development should take one-tenth the time that is required by third-generation languages. Is this a realistic expectation?

The general answer to this question is that it depends. It depends on several things, including the result that you are trying to produce and the person who is doing the work. But the fact that the answer to this question is not an unqualified "yes" does not mean that a ten-fold increase in productivity cannot be realized. There are many situations where this type of increase is being achieved, and it may be achieved in your situation if you are currently using third-generation languages. Even where there is not such a large increase in productivity, systems development with fourth-generation software moves along at a swifter pace than if third-generation languages are used.

I would like to explore some reasons why fourth-generation software products help increase programmer productivity. It is not my purpose to deliver a sales pitch that expounds the outstanding qualities of certain products. But when you look carefully at all of the features of fourth-generation software products, you may summarize many of the benefits into one single statement: fourth-generation software products allow you, in most cases, to describe what you want as the end result. You do not need to describe how to reach it.

For example, to print a page heading on a report, with fourth-generation software products you only need to describe what you want the heading to look like. You do not need to provide the logic that determines when and how it should be printed. Many of the time saving advantages of fourth-generation software products flow from the fact that it is much easier to describe the desired end result than to describe both the end result and the precise method needed to achieve it.

Productivity in the area of program maintenance is also closely tied to this characteristic. It is quite often the case that over the lifetime of a program, the cost to maintain it can exceed the original development cost. Therefore, the expense of maintaining systems should also be considered when evaluating the usefulness of fourth-generation software products.

A major problem with maintaining programs written in a third-generation language is that making a change can often result in damage to another part of the program. Altering the logic of a program increases the probability that the program will have more problems after you have fixed it than before you started. The

use of fourth-generation software products can virtually eliminate this problem. Once you have described the change in the end result, it is up to the fourth-generation software to make the appropriate adjustments to the logic. Program maintenance can be another big area of savings when using fourth-generation software.

But saving money by using fourth-generation software products to develop systems only makes sense if the application can be programmed using the fourth-generation software. The current level of sophistication of fourth-generation software does not make their application suitable to all situations. You may have personally come across a situation where you have had to fall back on a third-generation language.

In my experience, however, for every situation where you cannot use fourth-generation software, there are probably several situations where you can use fourth-generation software to satisfy needs that would otherwise remain unmet. An ad-hoc report which requires only five lines of code in a fourth-generation language can give you data that you would probably never even try to obtain using a third-generation language. You may be unable to use fourth-generation software for a specific application. but, without this software you give up the benefits that can be realized for less complex situations as well as the added dimension of ad-hoc reporting.

While we're looking at reports you can and can't write with fourth-generation languages, I would like to propose a valuable new feature. The fourth-generation languages that I am aware of allow access to only one record in a file at a time. But the ability to have access to data from a previously read record would greatly enhance the power of fourth-generation software. For example, consider a system which tracks the movement of equipment. If you could compare a record with a previous one, it would be easy to detect illogical sequences of moves. But detecting these illogical sequences is virtually impossible if you can only "look" at one record at a time.

As with third-generation languages, the person using fourth-generation software products can obtain better results if he understands the product well. In fact, in certain respects the user's knowledge of the fourth-generation software must be more complete than the programmer's knowledge of the third-generation language. A COBOL programmer who knows the basic language statements can usually be creative and produce the desired results. However, if a person using fourth-generation software is unaware of the particular option that will allow him to easily complete his task, the task may have to be abandoned. For example, a COBOL programmer may use the same basic statements

which produce page headings to put an initial heading on a report. A person using fourth-generation software would find it impossible to produce that same initial heading if he were unaware of the initial heading option.

I am not saying that a fourth-generation language is harder to learn than COBOL, or that you can't achieve complex results with fourth-generation software. But you need to be very familiar with a fourth-generation product before you can expect to produce certain things that at first sight seem to require a third-generation language.

A good data dictionary is a tool which can boost the productivity of all users of fourth-generation software. The data dictionary should allow the database designer to specify value edits, default column headings, help messages, COBOL-like picture clauses and ther options for the individual data elements. Once these have been described in the data dictionary, the effort required by users to produce attractice reports and useful data entry screens is greatly reduced.

Database design is an important factor in the efficiency of any system. A well-constructed report program can be terribly inefficient if it reads data from a poorly designed database. When a database is being designed, it is difficult to foresee all of the future reporting requirements. Therefore, with any database, you may at sometime find yourself having to write an inefficient report program. But with fourth-generation software, you may be able to speed up such a report with what I will call an ad-hoc file.

Consider a situation where you need to match records from two files according to a common key as well as a non-key date field. You would normally read all of the records that match the desired key value and then select from those the ones that have the correct date. If you are reading from a database which holds a great deal of historical data, this method would be very inefficient. However, this would probably still be the best method using a third-generation language.

If you were using a fourth-generation software product you could take advantage of the ability to easily produce ad-hoc files. In this situation, an ad-hoc file would work as follows. You would create a simple MPE sequential file of the records from one file with the desired key values regardless of the date. The next step would be to FCOPY this MPE file into a KSAM file with the date now included as part of the key. Then, using the fourth-generation software, you could read through the other file and match records from the newly created KSAM file according to the original key and the new date key. Now you read only the records

that you are interested in.

Before you write this off as too inefficient, you may want to try it. In a situation similar to this, the use of an ad-hoc file reduced the run time of a report to one-sixteenth of its original run time. Creating and using an ad-hoc file is much easier with fourth-generation software than with a third-generation language.

This example is difficult to follow and may not make much sense if you have not had to face this type of problem before, but don't despair. The general guideline that I would like to draw from this example is much simpler. When using fourth-generation software products, be creative in using their capabilities to create ad-hoc files and allow simplified access to different file types.

Before I conclude, I would like to deal with a major area of criticism concerning fourth-generation software products. Some people argue that applications using fourth-generation software run too slowly. Depending on the specific package and the specific application, this can be a valid complaint. Benchmarking fourth-generation software packages against third-generation languages is far beyond the scope of this paper. However, I would like to state a few thoughts that you might consider if this is a problem area for you.

Numerous studies have shown that it can cost more to develop a program than it will cost to run it over its entire lifetime. In this situation it seems important to consider the development costs of a program in relation to the run-time expense. If a program takes ten percent longer to run but costs one-tenth as much to develop, you may still be saving money.

Some critics of fourth-generation software products have taken an integrated approach. Without giving up the benefits of fourth-generation software for formatting data for reports, you may find it desirable to use another method for data retrieval or sorting. An example of this approach would be to retrieve the desired data and then use a sorting utility which is faster than the standard SORT/3000 usually used by fourth-generation software products. You could then use the fourth-generation software to read the sorted data and produce the report.

The ability of users to obtain results in one-tenth the time with fourth-generation software can be very beneficial to any organization that has data processing needs. Even though you will not obtain such spectacular productivity increases in all situations, many people have found that the use of fourth-generation software products is a very effective

way to solve problems. The beneficial results that are expected from fourth-generation software products are, in fact, being realized and in many cases are being surpassed.

A large group of people are currently using fourth-generation software products on the HP/3000. Their numbers are increasing and will no doubt continue to increase. The growth of fourth-generation languages is starting to follow the same pattern experienced by third-generation languages. Just as third-generation languages became more popular than second-generation languages, so the new fourth generation of languages may someday gain a strong position relative to third-generation languages.

User Documentation: The New Approach

By Sanford Rosen, Ph.D.
with Sara Beck Rosen
and Susan Furlow
Communication Sciences, Inc.

Since the beginning of the computer industry, application user documentation has had a basic responsibility to fulfill. That responsibility has been to "make it work." In order to "make it work," documentation must, clearly and quickly, transfer information from the people who understand how something works to those who have been assigned the task of learning how to make it work. On the surface, this appears simple enough. However, history has proved this kind of transfer of information to be enormously difficult.

Up until now, paper-based manuals have been the major medium used to accomplish this transfer of information. However, the problems associated with their use have been extremely difficult to overcome. What we've learned is that it's not simply a matter of codifying the documentation itself. Rather, it turns out that paper-based manuals, no matter how immaginatively packaged, inevitably intimidate the uninitiated user. And if you are lucky enough to seduce the user into becoming actively involved with a manual, whether or not the "make it work" information contained in the manual gets successfully transferred to the user depends on many things. For example, one of the most difficult things to guard against is the user's "wall-stare." ("Wall-stare" is a reader's disease, related to boredom and a perceived lack of "need-to-know.") Although many of the communication problems related to wall-stare and other reader/text interactions have been solved by creative technical writing groups, the computer industry still needs a mainstream solution to the problem of transferring "make it work" information, and preferably a solution that considers the problems of the user community.

We must not minimize the communication problems that now exist in the computer industry--and for that matter throughout our society. The United States is rapidly shifting from a mass industrial society to an information society, and the final impact will be more profound than the 19th century shift from an agricultural to an industrial society. In connection with the shift to an information society, it is important to notice a powerful

anomaly developing: as we move into an era in which the literacy of the general public is more important than ever before, our schools are turning out people whose general level of literacy is becoming progressively lower. SAT (the college entrance exam) scores have been going down for more than a decade. Consider this: for the first time in the history of the United States, the generation that is graduating from high school today is less skilled than its parents. This fact compounds the already difficult problem of transferring information.

But there are other problems involved in the successful transfer of information. One of the biggest of these is the failure of people in the computer industry to realize the absolute cruciality of documentation. Those at the top have consistently ignored the fact that a link must be forged between computers and people, a link which for the most part is now missing. That "missing link" is documentation.

But how did an industry with such technical power allow a thing like documentation to become the missing link? And why, after 25 years of continued, rapid, technical advancement, has almost nothing been done to correct the serious problems caused by this missing link? In the first place, it has been difficult to convince even the most intelligent and progressive management of the dollar value of paper-based user documentation. Management has heard experts give academic and theoretical arguments praising the value of good paper-based documentation, arguments allowing that while it takes time and money to create documentation that supports the user, rather than gets in his or her way, the benefits of doing so can more than make up for the costs. Customer-support costs can be considerably lowered. User satisfaction can be increased, resulting in fewer returns and more repeat business. The documented product's image and reputation can be tremendously improved. Management has also been reminded, over and over again, that the user manual has other kinds of lasting value. It is an excellent training tool for any client, analyst, or programmer who needs to be familiarized with a system. It can teach, demonstrate, motivate, and remind.

The preceding arguments sound good to me. They should; I have been presenting them (along with many others) for the past seven years. Given what has been the state of the art of documentation up until now, these types of arguments have represented our best methods of solving documentation problems. However, even if we assume the existence of both good user documentation standards and an effective documentation methodology within which to apply them, management can rebut our best arguments with some very basic, embarrassing questions such as:

1. Yes, but will they read it?

2. Some users have limited technical backgrounds and meager reading skills. What guarantee do I have that documentation will do them any good?

3. Systems change. How can I be sure that user documentation is updated on a timely basis, disseminated correctly, and included and used in new versions of user manuals?

4. Can a manual be created that users can read without having to skip, jump, detour, and reverse their way through it?

5. What will a good manual save in employee time?

All technical writing groups that expect to stay in business, whether they be internal or free-lance, are familiar with these arguments and prepared for their rebuttal. This does not diminish the validity of management's questions, and it certainly does not eliminate the subjective nature of paper-based documentation and the unpredictability of its success or failure. Examining the medium of paper more closely, I find there are even more areas that contribute to the failure side of the scale. For example:

Organization. A paper-based manual must take on a physical structure of some sort. That structure must incorporate some standardized way of presenting information, such as in terms of:

- Tasks or functions.
- Responsibilities by audience group.
- Categories of information.
- Sequences or chronologies of information.
- Levels of detail (general to specific).

A manual's structure also determines such interdependent elements as how its sections and subsections are marked, how its table of contents is set up, how things like dividers are used, and so forth. Whatever decisions and choices one makes about these structural elements, the resulting manual cannot possibly encompass all possible audience needs. The best it can do is to address the majority, if there is one. Since some choices necessarily negate others, this structuring, so fundamental to the paper-based manual, will also, in some ways, limit effective access to the information. Thus, no paper-based manual can even approach the utopian goal of being all things to all readers.

Updating. The process of updating paper-based user documentation is difficult, time-consuming, and expensive.

Consequently, those in charge of the process must institute an effective updating program that provides such things as:

1. An effective way of notifying the documentor of system changes.

2. An effective way of using word processing equipment to handle text changes, and possibly changes to graphics as well.

3. An innovative way of dealing with cross references.

4. A way of physically packaging the documentation so that pages can be replaced or added with ease.

5. A coherent system of recording and handling different versions and revisions, and of tracking who uses what.

6. An effective way of soliciting comments and update requests from the user community.

A comprehensive updating program of this nature requires commitment, not only from the personnel responsible for the tasks associated with the actual updating itself, but also from the people in management who are responsible for the budget required to support these tasks. However, despite the fact that it is getting more and more difficult to sweep the subject of documentation under the rug, technical writing managers continue to report that they have trouble getting the budget and scheduling allowances they need. The road that has led documentation to its current state is paved with good intentions and reduced budgets. Management has yet to learn the lesson of software economics: it costs more to monitor and modify documents than to develop them in the first place. But even if there were no budgetary constraints on the updating process, it turns out that there is at present no really efficient, cost-effective way of handling the six factors I've just listed. The standard approach to updating, as well as the standard approach to the whole question of documentation itself, must be reexamined. In both cases, methods must be developed that are more responsive and less expensive.

Apparency. "Apparency" is a word I have coined to describe a manual's overpowering physical presence, and the aspects of that physical presence that create in the reader an inescapable feeling of being overwhelmed. No matter how you package it, the paper-based user manual that can adequately support a sizeable application is itself sizeable. The total manual is physically "apparent" to all who must use it. The technical writer can employ all the guile available to him or her; he or she can color-code, use graphics to support the text,

construct a comprehensive index, use tab dividers for easy access, and develop psychological games to entice the reader. These techniques, as ingenious as they may be, cannot reduce the apparency of the user manual, and the intimidation that goes with it. This intimidation produces "reader resistance"; it reduces a reader's desire to read the manual, and usually lowers the grade level at which he or she reads, understands, and absorbs the information being presented. As long as the industry relies on paper-based documentation, the apparency problem and the resulting reader resistance cannot be overcome.

Revelation, Drama, and Interaction. These three elements must be present to keep the attention of any audience, and it's difficult to achieve any of them via a paper-based manual. The paper-based manual cannot adequately time and control the rate and impact of the process by which information is revealed, nor can the reader "interact" with it in any meaningful way. A paper-based manual cannot help but be a static and unresponsive object, in spite of the ways the technical writing community has tried to deal with this fact. And try it has. All sorts of imaginative approaches have been brought to bear. For instance, the "programmed text" is an example of a paper-based document that attempts to time and order the revelation of information in a way that is both dramatic and personally interactive with each and every reader. But the programmed text is not an appropriate vehicle for the kinds of information that go into most documentation.

You may think my use of the words "revelation," "drama," and "interaction" are a bit theatrical in the context of documentation. It is my contention, however, that we must again borrow from the arts to achieve our goal of the successful transfer of the kinds of information we're dealing with--"instructional literature," with a high potential for producing disinterest, boredom, or confusion in the reader. I say "again" because technical writers have borrowed from the arts before. I refer to their use of the "playscript" format. This format, which is akin to the script of a play, uses role assumption to give the reader a feeling of active participation in whatever the documentation is describing. However, we could go even further. With today's technology, we need no longer rely on the paper-based approach. Instead, we can introduce more imaginative, creative, and engaging methods of achieving our goals, as I will explain later in this paper.

The reason we must do something soon is because the problem is becoming more crucial in direct proportion to the rate at which technology is evolving. Here is what is happening:

1. Hardware costs are coming down.

   - All that technology is getting crammed into smaller and smaller packages, and being produced in larger and larger quantities at lower and lower costs.

2. The user population is expanding.

   - Sophisticated products are cheaper and thus available to more people--and more people are buying them.

3. User sophistication is decreasing.

   - The expanding user population includes more people who are not trained in the product technology.

4. Traditional methods of supporting a new purchase are becoming less effective.

   - Twenty years ago, when products cost $100,000 a throw, a manufacturer could afford to assign a full-time representative to the user organization. Today's purchase price cannot justify expanding a service organization so that it is big enough to give advice to all of the users all of the time.

Just as the traditional methods of support must be abandoned, so the traditional paper-based approach to documentation must be abandoned in favor of a new approach--one that can begin to unite the various functions served by documentation, training, and education. This new approach must not only overcome the reader/text interaction problems of the past, but must also take advantage of the technological advances available today. It is my firm belief that this new approach must be one that uses automated documentation methods. By "automated documentation methods," I mean a collection of standards, examples, and instructions that are on-line, and are able to be called up, in various configurations, by anyone doing documentation. This will all but eliminate the need for paper-based documentation with its inherent problems, problems which have helped obscure the importance of documentation in the minds of managers whose failure to appreciate this importance may have been fed by the subtle realization that no documentation they've been exposed to so far has been totally adequate. This new approach, if formulated carefully, can lead the computer industry into a new era--one in which the user community will be in possession of tools and techniques that will allow it to assume a new level of responsibility for solving its own information-transference problems, and for tailoring its documentation to its own needs with a specificity never before possible.

The concepts of computer-based education and on-line "help" to support application software are not new. Many software companies and data processing departments have used "help" screens and computer-based tutorials to assist paper-based documentation in the transfer of information. However, the approach presented in the remainder of this paper advocates the establishment of a new relationship between application software and what I call a "documentation/help subsystem." The design and implementation of such a subsystem must (as stated earlier) take advantage of advanced technology, incorporate the newest automated tools, and be reponsive to the needs of the user community. A fundamental requirement of such an on-line documentation/help subsystem is that it be designed as a stand-alone system, rather than be included as routines built into specific applications ("applications" here meaning, e.g., accounts receivable, accounts payable, order entry, manufacturing, etc.). The programming call for "help" would then be made to the stand-alone documentation/help data base, instead of to a routine in the application code. This approach would permit members of the user community (i.e., writers, documentors, readers, editors, etc.) to create, modify, and maintain the documentation/help subsystem data base without disturbing the existing application code.

Whether or not the users succeed in their attempt to assume the responsibility for the preparation of their own documentation will depend on the nature and quality of the tools and techniques supplied to them by the stand-alone documentation/help subsystem. To begin with, such a subsystem must contain a data base of on-line user documentation standards. These standards should present generic patterns for the users to follow when they prepare on-line user documentation. The standards should use a "boiler-plate/fill-in-the-blanks" approach to assist uninitiated users through the path of least resistance. The users should be able to add standards as they become more and more familiar with the subsystem and with their own needs. Examples of the categories of standards the initial standards data base should include are:

- How to Use HELP.
- A System Overview.
- System Conventions.
- User Procedures.
- System Screens.
- System Maps.
- Error Messages.
- Files/Data Bases.
- Glossary/Index.
- How to Use Your Equipment.

The inclusion of such standards, supported by a data base of examples, will help the user avoid the blank-page

syndrome. Questions like, "How do I write on-line documentation?" and "Where do I start?" can be answered immediately. Once the user has prepared on-line documentation/help to support an application, he or she must have the capability to review that documentation and the standards. Keeping standards, examples, and previously prepared documentation stored on-line will result in a library of solutions that can be applied to future application user documentation/help. This will eliminate some of the trial-and-error associated with the preparation of most paper-based documentation.

Other user-assistance tools aimed at increasing the effectiveness of the documentation should also be included in the documentation/help subsystem. For instance:

- A library of graphics symbols to be used to supplement narrative where necessary.

- A general glossary of industry terms and definitions to help guarantee that such terms are used consistently.

- A data base of general writing standards that can be adopted by the company as a whole.

- A number of note pads that people can use for communications directly related to the documentation. The design of these note pads can vary according to the various levels and functions of the people using them--for example:

  - Researchers.
  - General readers.
  - On-line application operators.
  - Writers.
  - Editors.

Functionally, the documentation/help subsystem must be data base-oriented. It should combine menus and commands for user efficiency. It should interface with an automated data dictionary, if one is used, and be capable of creating a text file that can be manipulated by a word processing system. This data base orientation will support the most effective updating--especially when documentation/help is standardized. It will also reduce the storage requirements, since no physical pages of it will need to be stored. Screens can be made up of logical pages, assembled only when being viewed. The data elements information may be stored on the automated data dictionary, if it allows for user definitions, and if it is being used consistently across applications. Boilerplate standards and company-specific standards should be on the standards data base, and exception data on a document data base. To efficiently access these data bases, the user must be supplied with an effective method

of navigating within the documentation/help subsystem. Therefore, I recommend that system navigation be accomplished through the use of a combination of menus and/or commands. This approach accommodates both the uninitiated and the experienced user. Supplying the user with a word processing capability--through the manipulation of a text file--allows for the preparation of paper-based documentation (and possibly graphics) on an exception basis.

The following narrative and graphics present a simplified overview picturing a possible structure of a documentation/help subsystem. The subsequent information breaks down the two components (A and B) to further illustrate some of the subsystem's features and functions.

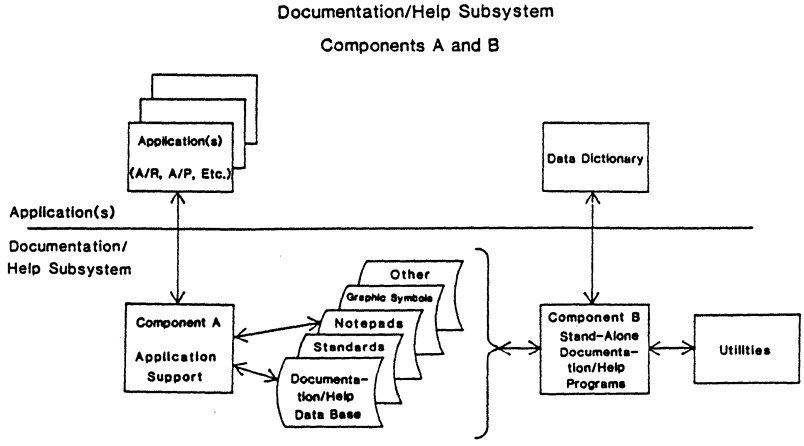


figure 1

Figure 1 defines the relationship between the application(s), the automated data dictionary, and the documentation/help subsystem components A and B. Component A directs the calls made to the documentation/help subsystem from the application(s). Component A also returns control to the application when the user/operator exits the documentation/help subsystem. Component B is the stand-alone portion of the subsystem that enables documentors, readers, editors, etc., to create and update documentation/help on the data base that is accessed through the use of component A. The personnel who use component B will have a wide range of tools available to them, such as: note pads, graphics symbols, data processing documentation/help standards, company-created standards,

general writing standards, word processing and sort utilities, and a reformat/print facility. The application user who uses component A to call the documentation/help data base will find many levels of information presented in a structure that makes navigation very simple.

Component A is used through a combination of menus and/or commands. This will allow users to call the documentation/help data base to get information that is outside the normal capabilities of the single application screen being viewed. Once into the documentation/help data base, the user may get to menus and directories, or, if the commands are known, navigate around in the documentation, viewing data presented in any of the categories shown in figure 2.

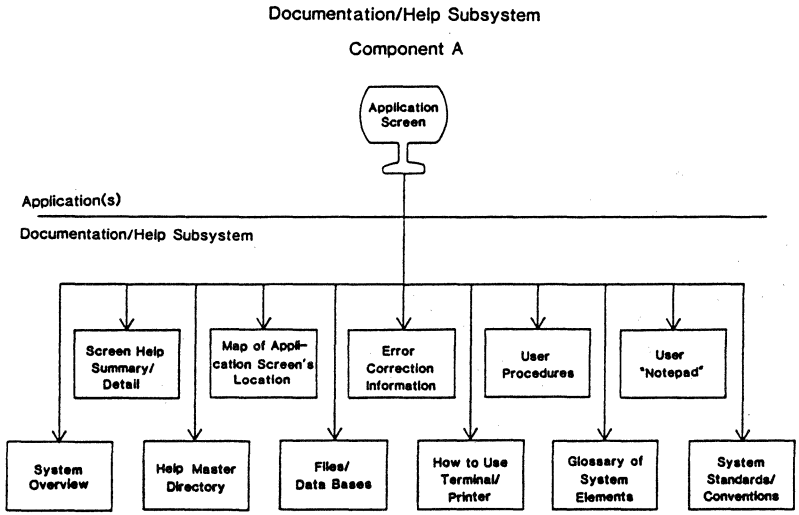


figure 2

The application user will be able to read any support screen and, if necessary, make notes on his or her personal note pad. However, since the application user will have entered the data base through Component A, he or she will be prevented from making any changes to the documentation data base itself. The right to manipulate data on the documentation/help data base is protected, and may only be accomplished through the use of component B. Component B is, of course, controlled by security identifications granted to documentors, editors, etc.

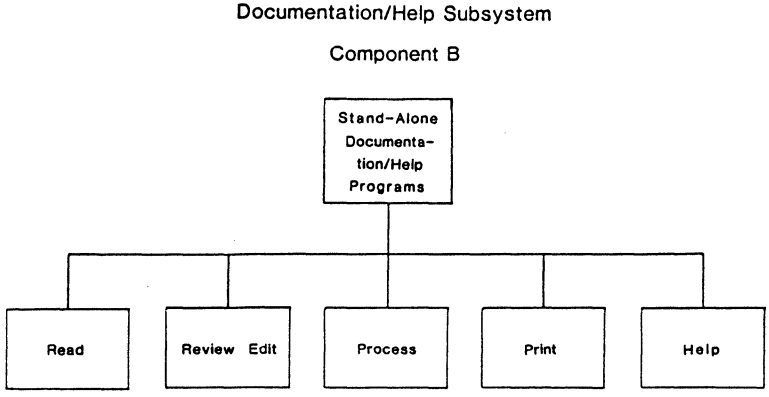


figure 3

Component B (figure 3) is the functional "guts" of the documentation/help subsystem. It has five major functions within which reside the important subfunctions needed to create, update, and read the documentation/help data base. The "read" function allows users to read documents without having entered the subsystem by way of component A (an application). Thus the system can accommodate any kind of documentation, such as programming, operations, or whatever you can imagine. The "review/edit" function allows reviewers and editors to read and comment (by use of notepads). The "process" function allows a writer to create a documentation/help data base by building it from scratch, by assembling it from pieces of existing documentation, or by using a predetermined boiler-plate standard. The "print" function converts information located on the documentation/help data base into a text file. This file can then be manipulated by word processing software. Finally, the "help" function answers user's (writers, editors, etc.) questions just as they would be answered if the user used component A to get to the documentation/help data base for application support.

Structuring the documentation/help subsystem with the stand-alone approach presented in this paper offers the user community (or whoever assumes the responsibility for preparing user documentation) a level of flexibility that encourages experimentation as part of the normal preparation of on-line user documentation. Consequently, the way information is constructed and presented will be varied in response to the signals (direct and indirect) from end users themselves. This approach will help build reader confidence, and promote a refining of the tools and techniques that the documentation/help subsystem

introduces. Correspondingly, many of the problems from the past can be left in the past.

The problems associated with paper-based structure and organization can be all but eliminated because of the logical rather than physical assembly of information. Updating, and the disseminating of new or modified documentation/help will be done with the assistance of a data base, and in an on-line environment. This will offer the most responsive updating ever, and should mean more effective use of computer software--which, today, has survived in a relatively unresponsive environment.

The problems connected with the "apparency" factor associated with paper-based documentation will also be alleviated. With the exception of some supplemental paper, all documentation will be totally transparent to the user. The intimidation factor will be gone. The information will be revealed to the user as he or she needs it. The overall physical volume of the documentation/help will be hidden. The documentation/help will unfold to assist the user as the need to know is communicated to the subsystem through an interactive process between user and documentation. The on-line documentation/help subsystem will bring revelation, drama, and interaction to the process of transferring "make it work" information to application users.

I also perceive that this new approach will begin to put an end to the dreaded reader's disease "wall-stare." Users will be able to more quickly find their answers and return to application functions. This will eliminate the normal page-leafing and index-searching that goes with a thick manual and that inevitably generates the need to rest, and the desire to stare for awhile at the nearest wall.

References Sources

- "The new Economic and Political Order of the 1980's" Vital Speeches of the Day by John Naisbitt. Foresight Group, Stockholm, April, 1980.
- "Responsible Documentation" Computer World, January, 1982. Neal Margolis.
- "Death, Taxes, and DP Documentation" Datamation, February, 1981. Lindsay Wilson.
- "Usability: Toward a Science of User Documentation," Computerworld In Depth, January 1983. Edmond H. Weiss.
- "User Manuals Insure Client Involvement," Journal of Systems Management, June, 1979. William C. Frank.
- "Tech Writer Talk," Datamation, February, 1981. Perry Petersen.
- "EXPLAIN," A writing/documentation methodology. Communication Sciences, Inc., 1982.

USER FRIENDLY SOFTWARE DEVELOPMENT

by

Ivan M. Rosenberg
Distinctive Solutions Corporation

I. INTRODUCTION

This paper discusses the characteristics of "user-friendly" software systems, and proposes development methods and techniques which can efficiently produce such a system.

WHAT IS A "USER-FRIENDLY" SYSTEM?

It is important to first have a good idea of the intended result before discussing how it will be created. What does "user-friendly" mean? Often it refers only to the resultant software system, in particular the ease with which a user can direct the system to perform desired tasks. A system that provides the ability to select from menus is usually considered more "friendly" than one that requires the entry of one of a possible thousand codes.

However, what if the menu trees are so "deep" that it requires a significant time to get to the menu that contains the desired selection? The frustrated user who knows exactly what he wants but has to labor through twenty menus will not feel that the system is very "friendly". He would prefer to specify which function is to be performed immediately upon system entry. In this case, a "keyword" approach may be preferable.

Thus, "user-friendly" is not a quality that is inherent in the software system itself, but rather is a term that can only be applied to the user-system combination. The characteristics of the user must also be considered in deciding whether a particular system is "user-friendly".

We must also realize that it will be the user who will make the "user-friendly" judgement, not the data processing professionals. Thus we must look at what the user may consider in making this evaluation.

THE USER'S VIEWPOINT

What may be surprising is that users, consciously and subconsiously, consider more than just the resultant system in their evaluation, although that is a prime determinent. In our experience as application software providers, users appear to focus on four dimensions: Quality, Functionality, Cost, and Time.

> QUALITY: the "degree or grade of excellence" of the system, often related to the reliability, consistency, ease of use, and support.
>
> FUNCTIONALITY: the degree to which the system provides desired application-oriented functions, including specific computations, reports, and inquiries.
>
> COST: the resources needed to receive the perceived and desired benefits, including monetary funds, personnal, hardware, and training.
>
> TIME: the elapsed time required to achieve the desired results, including time from commitment to production, and the time required by the system to perform its functions, including response time.

There are tradeoffs between these four dimensions. To the extent one optimizes one dimension, the other dimensions tend to be "suboptimized". For example, the more functionality demanded of a system under development, the more the product will likely cost and the longer it will take to be installed. The more complex a system, the more execution time will likely be required.

DEFINITION OF "USER-FRIENDLY"

It is up to the developer to make the user explicitly aware of the tradeoffs, to appreciate the user's weighting and considerations under each dimension, and to propose and create a solution that optimizes the user's weighted average.

This result in the following proposed definition for a "user-friendly" system:

> Given a specific user, a "user-friendly" system is one which optimizes the user's weighted average evaluation of the system over the dimensions of quality, functionality, cost, and time.

OUR BACKGROUND

Distinctive Solutions specializes in providing software products to banks and commercial finance organizations. This marketplace is characterised by new users, a requirement for a high degree of reliability, the need for exception and inquiry information in a timely manner, ad-hoc inquiries. personnel turnover, changing requirements as as marketing approaches change, and a labor intensive operation. The limiting factor in a financial organization is often the ability to process information within required timeframes.

In the past, many such organizations developed their own custom system. However, the cost of maintaining these systems and keeping them up-to-date with the emerging computer technology and the needs of the users has increased the demand for standard products. Yet, despite the common aspects of the application, we have found that each company is quite different in the terminology and procedures it uses, in the data it feels are important to monitor, and in the specific computational algorithms it uses for computing interest and other fees.

The challenge for our company was to develop a standard product for the commercial finance industry that would be evaluated as "user-friendly" by a wide range of potential customers. The product had to be easy to customize and enhance, easy to learn, had to be supported as standard product, extremely reliable, and had to be sold and maintained for a price that was both cost-justifiable to a sufficient portion of the marketplace and profitable for us.

This paper is a result of the experience we have gained in achieving this goal.

OVERVIEW

The rest of this paper is divided into four sections:

II. THE SYSTEM LIFE CYCLE: which is used as a context for discussing software development methods.

III. SOFTWARE DEVELOPMENT METHODS: which discusses development methods that lead to user friendly systems.

IV. CHARACTERISTICS OF USER-FRIENDLY SYSTEMS: a set of proposed design and implementation characteristics that promote a feeling of friendlinees in users.

II. THE SYSTEM LIFE CYCLE

Every system, whether automated or manual, has a life cycle: a beginning, an existence, and an end. The life cycle may be viewed in terms of phases, just like a person's life may be viewed as beginning with the decision to have a child, and proceeds through gestation, birth, youth, maturity, and death.

Since the user is one of the "parents" of a system, it is important that the user have a context for understanding how the "child" will grow. Understanding the complete life of a system is also important in insuring that the right foundations are laid. Too often the focus is on getting the system running, rather than on the steps that are necessary to insure that it will run correctly. For example, there is often pressure to begin programming early in a development project, instead of insuring that the design (which defines what will be programmed) is complete and is actually what the user wants. If users fully appreciated that the programming is completely dependant on the design phase, there would be fewer costly changes to a system soon after it is installed. To a certain extent it is like the car repair man in the TV ad, "you can pay me now (to do the design right), or you can pay me later (to fix the program)". Since it is much more expensive to "pay later", and the user is often the one who does the paying, it is important that the user takes responsibility for insuring that the design, and in fact all the phases, are done thoroughly and correctly. Thus, it is important that the software developer educate the user early in the project concerning the System Life Cycle so the user may be a fully responsible member of the development.

The phases of the System Life Cycle are:

1. FEASIBILITY PHASE (Parent's Decision): This begins with the determiniation of whether there is a problem and, if so, its definition. This step is often given little formal attention, yet it provides the foundation for everything which follows. This phase ends when it is formally decided to try to solve the problem, and approval is given to the proposed Solution Process.

2. SELECTION PHASE (Conception): Once the problem is identified, this phase includes the analysis of potentially feasible solutions and the selection of one. This phase tends to have the most managerial involvement.

3. DEVELOPMENT PHASE (Pregnancy and Birth): This phase brings the solution into being. In some cases the system already exists as a product that can be installed without modification, in other cases the system must be developed "from scratch", and in still other cases an existing system will be modified prior to installation. This phase requires full user participation to insure the desired results.

4. INSTALLATION PHASE (Youth): This phase covers the time from when the system solution comes into being, through installation and acceptance of the new system, including data loading and training. The last activity of this phase is usually the formal switchover from the old to the new system.

5. PRODUCTION PHASE (Maturity): This phase covers from switchover, through maintenance and enhancement, including the years of useful and productive work.

6. REPLACEMENT PHASE (Death): At some time technology and needs grow sufficiently to make it cheaper to replace a system than to do further modifications. This phase includes the replacement of the system by a new system, and the final shutdown of the old system. This phase for the old system overlaps the first four phases of the new system.

There is nothing that is particularly "computer" about the System Life Cycle. It is true for every project, whether installing a new computer or opening a new branch office. However, it is extremely important that the project manager, one of the critical components to success, be a skilled project manager with good support.

III. SOFTWARE DEVELOPMENT METHODS

1. PRINCIPLES

There are some general principles to software development that underlie all the specific methods and techniques that we have used. These principles set the tone for the project, they form a set of guides that can be used to check whether day-to-day activities are consistent with the desired results. These are not necessary revolutionary principles; often they are just common sense. But unless they are explicitly stated, adopted, and frequently reviewed, the issues of the moment will almost always dominate the long-term view.

These principles are very attitude-based rather than specific methods. Engineering techniques, such as structured design or PERT charts, are just the tools. The most important component of a successful relationship is the context of that relationship, and that is what these principles address.

A. RESPONSIBILITY: Everyone, especially the user, must take full responsibility for the success of the project. If everyone is responsible for every aspect of the project, then there is less time spent on allocating blame and more spent on finding solutions.

B. USER INVOLVEMENT: The user must be involved throughout the Life Cycle. This is the only way the user can act on his responsibility for the result, and can influence the development process so the result is the one desired.

C. SYSTEM OWNERSHIP: The user must "own" the system. That is, the user must feel that it is "his" system, not that he has to adapt unwillingly to the system. This is a psychological perception that includes, but is not limited to, the system features. If this principle is not followed, then any undesired outcomes, even those that result from tradeoffs, will be blamed on the developer.

D. SERVE THE USER: If you are in the business of software development then one of the principles of success in the long term must be to serve the user. Sometimes this means explaining the same thing three times, understanding user concerns instead of being defensive, and generally looking to see what works.

E. BE PATIENT: This is often the toughest principle to implement. Software development is often a new experience for the user, and there is frequently a lot at risk; money, reputations, sometimes the company's future. The user is often anxious and afraid. The experienced developer has

been through it before - he knows what is trouble and what is not. It doesn't work to be defensive - the user will only become more concerned. Fundamentally the user just wants to know that the developer is competent and can handle things, and that the user will be taken care of. This can be implemented in a wide variety of ways: supplying extra data (relevant or not, as long as it is what the user wants) concerning the project, insuring the user gets all his concerns expressed, explaining the same thing multiple times (patiently), warning the user of anticipated issues, and educating the user in the development process.

It will help to put yourself in the client's shoes. Remember the first time you flew, or drove a car, or did something else that used to be frightening? Now you know what to expect and are casual about it. The user is flying for the first time and you are the pilot. Keep your passengers as comfortable as you can.

F. MAKE THE EXPECTED RESULTS EXPLICIT: Insure that the expected results are explicitly stated in writing, and that all project team members and management subscribe to the results. Often it is assumed that everyone knows what the expected results are, and it is too often not true. The importance of mutually agreed upon results is crucial enough that a project should not begin until this is completed.

G. MANAGE THE PROJECT: It is my perception that the primary cause of unsatisfactory results is inadequate project management. Project management is not something that can be casually assigned to the designer or lead programmer. The required skills are different.

For our major projects we assign a full-time project manager from the very beginning. Preferably this person does not also have an operational role in the project, such as analyst, designer, or programmer. It may be that there is time for this person to perform other tasks, but the primary responsibility should be to manage the project, from contract signing to Acceptance Test completion. To many software development organizations this probably seems like a very large overhead. However, project management must be done, and it will either be done inefficiently and show up in cost overruns and delayed deliveries, or it will be done efficiently by a specialist.

It is important to also manage the user as part of the project. A user's fears will often lead them to make demands that may not be consistent with the overall project goals. Insisting that programming begin before the design is complete is one of the most common examples. This is

where a skilled project manager will save both the developer and the user considerable time and money.

H. MAKE THE PROJECT A SUCCESS: If this isn't a principle then one shouldn't begin the project in the first place. Defining what constitutes "success", the expected results, is the first step. Being responsible for and committed to accomplishing the results is the second step. Actually doing what is required for success is the third step. This is where "results, not reasons", is a useful guide.

Making the project a success includes insuring that the successes are acknowledged. Often pragmatic engineers focus on what is to be done rather than on what has been accomplished. Project management will produce a set of specific milestones. These can be used to compare actual progress to the plan and acknowledge accomplishments. Insure that there is a frequent sense of completion instead of just moving on to the next task. After the system is installed and running successfully, have a party and publicly thank those who risked by authorizing the project, and those who worked nights to produce the results.

One of the best ways to promote success is to have an environment where success is expected. Visibility of milestones, acknowledgements, good management, and refusing to participate in mutual fault-finding all help. Another useful tool is the goal of making the project results even better than expected.

I. BE EFFICIENT: The above guidelines focus on the results, this one is concerned with how those results are achieved. We found that it takes a conscious effort to look at how something might be done more efficiently. Often the way we do something is closely tied with assumptions that may no longer be true.

J. BE GENERIC: This guideline suggests that one should look to how the specific may be generalized. This leads to systems that are easier to enhance and maintain, and permits future systems development to build on the results of previous systems. We try to insure that the same algorithm is not coded twice.

K. HAVE FUN: As with success above, if this is not a goal, then why do the project? When things look their darkest, it usually means that there is an important lesson to be learned. There can be pleasure and fun in the learning process, and there will ALWAYS be lessons to be learned.

2. FEASIBILITY PHASE

The internal data processing staff is often involved right at the beginning of a project; an outside vendor is not usually involved until the Selection Phase. It is important that the vendor realize what has happened during this first phase and do what is necessary to "catch up" in terms of both the relationship and the project deliverables. For example, even after a vendor selection has been made many users have not clearly stated the System Requirements, which should be a result of the first phase. The following assumes this situation.

A. MARKETING AND SELLING (PRE-COMMITMENT)

Much of today's software is marketed by the developers, usually technically oriented people. However, much of what closes a sale is not directly related to the software functionality. The beginning of the relationship with a user is critical - first impressions definitely impact all subsequent perceptions, including the "user-friendly" evaluation.

FORM THE RELATIONSHIP: In any significant purchase of software, users evaluate the relationship as much as, or more than, the software functionality. It is important for the vendor to realize this and insure that any user concerns in this area are correctly handled. Take the long term perspective instead of the short term sale. Realize that the relationship is one that both vendor and user will have to live with for years. The vendor should also be evaluating the user to determine if the relationship is one that will work. Signs of potential problems include: internal political warfare, lack of upper management support, a history of unsuccessful data processing relationships, unrealistic expectations, and an unwillingness to take responsibility for the project.

HONESTY AND INTEGRITY: Software developers are overly optimistic; users are often overly pessimistic. This pessimism only increases when promises are not kept, or "white lies" are found out. One of the guidelines we use in aiming for "No Surprises" is letting a prospective user know, up front, that the time from commitment to production may be as long as nine months. This permits the user to factor that into his pre-commitment activities and gets you, the vendor, an earlier closure with less crises. We have developed standard literature that lets a user know a typical development installation plan early in the process.

Above all, do not make promises unless you know you can keep them, and keep the promises you do make.

CREDENTIALIZATION: Software vendors have a relationship with users that is very similar to that of doctor and patient. Unless you credentialize yourself, your company and your product, there will be continual small upsets whose source will rarely be clear. Credentialization should take place as early as possible, and it must be continually checked to insure it is in good shape.

Credentialization may be done through references, demonstrated knowledge, quality of company materials, financial condition, and the marketing/selling techniques used.

FORM AND CONTENT: Throughout the relationship a user will be exposed to much more information that he can reasonably absorb. Thus, much of the success of the relationship not only depends on the content of deliverables but also their form. Sloppy looking presentations leave a definite impression, regardless of the quality of the content.

In our presentations we have one person whose sole purpose is to handle logistics, to insure that both the presentor and the audience are well taken care of and that the form of the presentation communicates that we are a high quality organization.

KNOWING THE APPLICATION: One of the best methods of credentializing, and of insuring that the system is user friendly, is to know the application. It is important to realize that users do not always know, or are not always able to clearly specify their needs. Not knowing automated system concepts, users do not know the potentials and thus often suggest just an automated version of the current procedures instead of taking advantage of the computer's strengths. Thus, it is important that the vendor understand the application so that he may guide the user to a successful result.

ESTABLISH THE PRINCIPLES: We find it useful to establish agreement on the above principles early in the project. We review each one in terms of the user's and our role, and insure we are in concert. This sets a context for the project in which specific issues may be resolved.

B. COMMITMENT

During this step a mutual commitment to achieve specified results is made, usually in the form of a contract.

CLARIFY THE COMMITMENT: If, during pre-commitment, you only promised what you knew could be delivered, then this step should be easy. It is still worth insuring that the mutual deliverables are clearly specified and are mutually agreed to. A good control is to insure that those who will be

responsible for delivering on the commitments agreed with the commitments. Too often a delivery date will be promised by marketing that does not have the support of development.

COOPERATION: When a relationship gets down to details there tends to be a focus on what can go wrong, and who will do what to whom if it happens. While we do not advocate abandoning reasonable mutual protections, our legal documents accent the conditions necessary for a relationship that works. It mentions being prepared for meetings, having meetings start on time, and the deliverables of the user (personnel, design reviews, etc.) as well as those of the vendor.

C. FORM PROJECT TEAM: The Project Review Team should be formed immediately after (if not before) Commitment. This team is composed of representatives of the user, the software vendor, and the hardware vendor. It is chaired by the user manager who is responsible for the success of the project. The job of the other members of the Team is to support the Chairman.

The first task of this team is to approve the Project Plan and confirm responsibilities. The purpose of the Team is to monitor actual progress and to take necessary actions to insure success. A user representative being the Chairman emphasizes the fundamental responsibility of the user in achieving this goal. We have found that monthly meetings of the full Team works well, with at least weekly contact between the user and vendor project managers.

The vendor should insure that the measures of progress are explicit, simple, and well-defined. These are usually based on the four dimensions described above: Quality, Functionality, Cost, and Time. Since the first two are difficult for a user to evaluate until after installation, the focus is often on the latter two during development. A vendor who tracks cost and time, keeps the user informed, and meets commitments is more likely to have a friendly user.

D. SYSTEM REQUIREMENTS DEFINITION

This document is a major output of the Feasibility Phase. It states, in a subjective and sometimes contradictory manner, why the user got involved with the project in the first place. The requirements should be specified before a vendor is selected, but often are not. This is one of the "clean-up" tasks that the vendor should insure is done.

In addition to giving an overall "vision" to the project, the System Requirements Definition is used to evaluate project success during the Project Auditing Meeting, which

takes place about six months after production begins. It is during this meeting that the formal judgment on the "user-friendliness" of the system is made.

3. ANALYSIS PHASE

The purpose of this phase is to produce the System Objectives Definition, a document which specifies the standards, functions, and data fields of the proposed system. The concepts of structured analysis and design are useful here, but are not the primary subject of this paper. Rather, I will present some of the techniques we use to increase the probability that the result is what the user really wants.

INTERVIEW EVERYONE: In gathering data, insure a wide variety of user personnel are used as sources, particularly those actually performing the tasks to be automated. Ask them what is working well, what is not and how they would change it. Avoid evaluation at this stage, just gather data. We have found that often data gathered in this manner is contradictory and needs to be made consistent.

BE EFFICIENT: Two techniques we use are a portable tape recorder and word processing. All data gathering interviews and meetings are recorded using a lightweight micro cassette recorder. This avoids laborious note taking and permits reviewing the interview later to clarify understanding.

We also make extensive use of word processing. GALLEY is provided to all HPIUG installation sites and thus is almost a common word processing system that makes it possible for us to provide all documentation in machine readable form to users. It permits us to quickly provide users with draft documentation and to easily make changes as the development proceeds. We strongly believe that documentation must be comprehensive, correct, readable, and current.

PROTOTYPE: For a user to learn how a system is proposed to work, a prototype works far better than copies of screens. We use a method that permits us to build a prototype as fast as the screen layout can be input to VPLUS/3000, and no programming is required. The prototype screens are then the ones used by the program during implementation, so prototyping involves no additional costs.

EDUCATE: As stated above, most users will propose a system design that is based on the current manual system. It is up to the analyst to open the additional possibilities of an automated system (this is one reason why application knowledge is useful). This a a tightrope. I have seen a situation where so many possibilities were suggested that

the user put the project on hold until his confusion could be resolved.

REVIEW EXISTING SYSTEMS: Taking a look at similar systems and talking with their implementors seems an obvious step for both the analyst and the user, but it seems to be rarely done.

4. DESIGN PHASE

The purpose of this phase is to produce the System Design Definition, a document which precisely specifies how the proposed system will work from a user point-of-view, including screen and report layouts, a data dictionary, operating procedures, design and implementation standards, and algorithm details.

We use word processing to almost automatically and efficiently transform the System Objectives into a first draft System Design. Using the principle of writing code only once, much of the design can be highly standardized.

Each option is defined by the following sections:

A. Description - a user oriented description of the purpose and operation of the option.

B. Operation - a description of how the option is initiated, including input parameters. Most of the time the entire function operation is completely specified by the term "Standard Operation". This insures the resultant system will have a consistent operation and any exception to the standards will be explicitly stated.

C. Program Operation - a set of detailed notes to the implementors, including algorithms and any advice from the designer.

D. Messages - specification of any non-standard messages.

E. Layout - the report or screen layout, a specification of whether the field is required, optional, or display only (if for a screen), and the correspondence between the report/screen field and the Data Dictionary items.

Due to the detail of this phase, it is often difficult to maintain user involvement. As with all deliverable documents, the user is required to sign off on the Design before it is used as the basis for the next phase. This is usually enough to insure a careful review. This sign-off also includes a precise specification of everything in the document considered confidential.

5. IMPLEMENTATION PHASE

During this phase the programs are written and tested. The following principles guide much of what is done at this phase.

STANDARDIZATION: As much as possible is standardized, particularly coding techniques, variable naming, and procedures. The design standards make it possible to have implementation standards. The cost of maintenance is a significant consideration during this phase.

The coding standards we use include the following rules:

- The REDEFINES is not used since it can cause problems if the data base structure (and thus the WORKING STORAGE SECTION) is changed.

- All branching must be explicit. If there are n alternatives, the branch must explicitly check for each alternative, with the n + 1 branch being an error condition.

- The PERFORM THRU and GO TO statements are not used due to the potential for control structure errors.

- All interaction with the file system (including IMAGE), VPLUS, any peripheral device, or any third party utility is through program or subroutine calls.

TEMPLATES: Where possible templates are created for standard option procedures, such as data maintenance. Since the control sections of programs are a major source of the most difficult bugs, this eliminates a significant portion of potential errors, thereby reducing costs and implementation time.

GENERIC UTILITIES: Wherever possible generic (application-independent) utilities are written to perform a program function. For example we have utilities to handle all errors, to display all messages, to handle traversing the menu tree, for security, for creating stream files, and for the on-line HELP functions. Creating these utilities means that building future systems will be that much easier and less costly.

DOCUMENTATION: Things will change. Word processing makes updating easier, but project management must insure that it is done. Out of date documentation or poor documentation control procedures can actually kill a project even if everything else is done well.

TESTING: In my opinion, this is one of the most neglected areas of software development. Practical, efficient,

methods of software testing are now being developed, but too rarely used in actual practice. Software reliability must be built in from the very beginning of the project, it cannot be "tested in". Formal test plans should be developed with user participation. Beware of using randomly selected data for then the errors will be randomly found.

USER-INVOLVEMENT: Even though users are not directly involved with the programming, there is still much for the user to do during this phase. Conversion, Data Entry, and Data Verification Plans must be developed. We also involve users in casual module testing to insure that no glaring mistakes have been made. It is important to caution against premature data loading, since data bases have been known to require changes right up to the final program implementation. We also use this time to finalize the Acceptance Test, a precise specification of the tests and expected results that will be performed after installation.

6. INSTALLATION PHASE

Although much work may have been invested up to this point in user-friendly characteristics, it is during the installation that a developer must be particularly aware, for this is the first exposure that most of the users will have to the system. Many will be apprehensive, some may even be antagonistic.

The major goal at this stage, beside having a technically smooth installation, is to convince the new users that the system is comprehensiable, easy to use, and will support them in achieving the results that they want. This has to be done slowly, with humility, and patience.

A preview "show" during the analysis and design phases, using the prototype, can ease the way for the actual system.

For new computer installations we recommend at least one month for "shake-down", during which the computer system management is testing its operational procedures and users are becoming familiar with the computer system. The games on the HPIUG Contributed Library are excellent for helping people get over their fear of the computer, learn how to use the terminal, and to develop an appreciation (quite subconsciously) of computer concepts.

We focus on making the installation as easy and fool-proof as possible. To the extent possible we use STREAM files to do everything (ADAGER is an example of a well-designed installation procedure). It is particularly important that the installation go well since this is a very visible time with high user expectations. Difficulties during this time

tend to lead to user anticipation of further problems later.

We also install with a "ready-to-go" demonstration system and data base. This permits an immediate demonstration and testing of the system, with immediate positive feedback for the waiting users.

Of course, allow plenty of time to solve unanticipated problems, and insure that the proper support people are available.

TRAINING: It is absolutely mandatory that the personnel be well-trained to use the system. I believe that most systems fail not for technical reasons, but for people reasons, and many of those reasons are related to inadequate training. People fear what they do not understand, and they will cause the demise of a system they fear.

Teaching is as much an art as programming, and talents in one do not imply talents for the other. A well-known software firm has traditionally hired school teachers for customer training and support, and taught them the needed computer conncepts and application skills.

Training requires a good training plan, good system support, and good documentation. Also, sufficient elapsed time must be allocated, since people can absorb information at a limited rate.

7. PRODUCTION PHASE

SUPPORT: During the Production Phase the major "user-friendly" needed characterisitic is good support. There will be problems - some trivial and some difficult. They will ALL appear difficult to the user. Treat each problem with respect, even though it may be difficult to appreciate the position of a new user who is totally unfamiliar with something that you have lived with for years.

Our Phone-In Consulting Service is very important during the early production stages. This is a 24-hour telephone consulting service. Of course, during off-business hours the calls are handled by an answering service who then contacts the "on-duty" consultant. The relatively low cost for a special telephone line and the answering service yield significant benefits in increased user feelings of security.

FOLLOW-UP AUDIT: One of the first tasks was to develop the System Requirements Definition. About six months after

beginning production a Project Auditing Meeting should take place to test whether the desired results were actually achieved. This is rarely done in our experience, which leaves the project implementors with a strong feeling of incompletion and frustration. We have found this feeling of incompletion the aspect that "burns people out" the most. People need a sense of completion before they can address the next challenge.

BUGS AND ENHANCEMENTS: Now also is the time for a good bug and enhancement reporting/fixing program to be in place. Users can understand that it may take time to fix a bug. They tend to be intolerant of having to call the same bug to your attention more than once. The complaints one hears are rarely that the bug wasn't fixed quickly (unless it was a major problem), but more often that the reported bug didn't appear on the next list of current bugs, i.e., it wasn't acknowledged.

IV. CHARACTERISTICS OF USER-FRIENDLY SYSTEMS

This section summarizes some of the design priniciples that we use that appear to support user evaluations of being "user-friendly". As I stated in the beginning, none of these can be adopted blindly - the specific user must be taken into account.

1. USER'S POINT OF VIEW

To the extent that the system design and structure, and the system terminology, reflects that familiar to the user, the system will be more readily adopted.

MODULAR DESIGN: A modular design, where each program performs one well-defined function, supports this concept. As an added benefit, this also makes maintenance simpler. Thus, to the extent possible each on-line maintenance program updates only one file or data set. Processing jobs perform individual, well-defined tasks, such as interest computation for all loans at a branch. We try to make all system tasks, particularly the processing tasks, match conceptually the structure of the current manual system.

MESSAGE SYSTEM: One of our utilities is a Message System, which handles all messages to the user, whether through the terminal or a job. These messages can be completely customized by the user to conform to the terminology and practice at each site. It is possible for each user to have his own set of messages. In addition, this system

will permit non-English language messages so the same software may be used outside the U.S..

VPLUS permits the screen labels to be changed without affecting the programs. The Code File permits codes to be customized to the user. The only aspect that is currently not easily customizable is the report labels, and standardization may soom permit that. When that is accomplished, all codes and text that the user sees that is non-numeric may be customized without programmatic changes.

2. CONSISTENCY

Consistency is one of the most important characteristics a system can have. This is encouraged by the use and enforcement of standards. Inconsistent operation will cause frustration and inefficiencies.

TESTING: Sometimes consistency can be a hard objective to obtain as the implementation requirements are considered. When an exception is proposed, we make a "test run" of the user training to see if we can explain the operation in simple terms. If not, the exception is not allowed. Exceptions are, in fact, rarely approved.

FUNCTION KEYS: The use of the function keys is often abused regarding consistency. If a function is to always, or frequently, be assigned to a function key (such as EXIT) then it should always be assigned to the same function key.

We do not use function keys for specifying application functions. Function keys have the limitation of having only eight alternatives available at each "level", which could lead to very "deep" trees and illogical groupings of functions, and makes the specification of parameters for the function request a complex process (the keyboard is used to enter data and the function key is used to request the function, both could be done through the keyboard).

We use function keys to specify non-application functions that are always available, have meaning for all application systems, and are consistent. Our current standard is:

F1: Return to the System Master Menu
F2: Return to the Subsystem Main Menu
F3: Back up one screen
F4: Refresh the screen

F5: Cancel a requested action (Panic Button)
F6: Print the screen
F7: Confirm a requested action
F8: Exit the system

FUNCTION CATEGORIES: Every function in our systems is in one of four categories, as follows:

DATA MAINTENANCE: on-line, affects the data base

INQUIRY: on-line, does not affect the data base

PROCESSING: batch, affects the data base

REPORTING: batch, does not affect the data base

On each menu screen, the options are grouped by function category so the user is clear about the capability of the option.

In particular we think the separation between processing and reporting is an important one. During the time a processing program is running, the system is at its most vulnerable (to system crashes, etc.). It is therefore desirable to minimize the processing time. Eliminating any reporting function, with its often included sorting requirement, makes a major contribution to minimizing this vulnerable time. Processing programs may then be optimized without regard to reporting requirements.

Reports should be able to be run as many times as desired, should be interruptable without consequences, and should be able to be run as of past dates. These characteristics are exactly the opposite of most processing programs, and thus it is illogical to combine them into a single unit.

Our processing programs do produce reports that indicate and summarize the actions taken by the program, but these reports are intended only for checking to insure correct program operation.

3. SIMPLICITY

As simplicity is the major tool in producing reliable systems, it is also a major tool in making those systems user-friendly. Simple does not mean unsophisticated. Quite the contrary, we have found that the simpler we make the user procedures, the more sophisticated the implementation must be.

For example, a user may specify his office indirectly through the logon user name. When he selects a report, the system automatically sets up the report request to include only those entities within his line of command and directs the report to the printer of the correct type that is associated with the user's organizational unit. A field on the report menu permits the user to override the output printer by simply entering the identifier of the desired printer location. Other menu fields, which can be set by

default, permit the user to specify sort order, report groupings, date, number of copies, output priority, a password for report access, and a file name for on-line report display. Normally the user does not specify these fields. He only does so if the associated option should be different than the default. The programming behind these features is quite sophisticated, although simple in concept. To the user, it is extremely simple, as we intended.

4. PSYCHOLOGICAL CONSIDERATIONS

The psychological considerations are sometimes the hardest to determine, before or after the fact. Here are some hints that have served us well:

FEAR OF THE UNKNOWN: We assume the user, particularly the new user is afraid - of looking foolish, of doing damage, of making a mistake. They have heard of the famous missing comma in the FORTRAN program that caused a missile crash. They imagine what they could do from a terminal.

Thus we try to build a system that is comprehensible and friendly. The training is designed to take the user step by step, and the system operation proceeds in the same manner, uncovering more detail as one proceeds to "deeper" levels. This is a typical characteristic of menu drivin systems, which is one reason most of our systems are designed on this basis.

We also respond to each and every user input. If there is to be a wait, we display "PLEASE WAIT" and the reason. A message is displayed whenever the data base is changed on-line, confirming that the desired action took place.

All error traps result in a "soft-failure", i.e., a specific message indicating the cause, severity, and occassionaly the recommended action. Although error codes are included for the data processing staff for fatal error messages, all errors specify the situation in readable text.

KNOW WHERE YOU ARE: The user should know "where they are" are every moment. Again, a menu tree structure can assist in this. Even within a program the user should be clear about his alternatives at every step and where those alternative fit into an overall application objective.

For example, a question and answer system can be confusing during data entry and correction when there is a provision for "backing up" and "leaping forward" over data items.

EXPLICIT DATA BASE CHANGES: Changing the data base is one of the most powerful user capabilities (as compared to

requesting an inquiry or report), and thus it must be explicitly clear when a data base change will potentially be made. For example, in our data maintenance procedures, the data base is updated only once for each "transaction", and that occurs when the user presses ENTER with an ADD or CHANGE in the Action field. Even on multi-form maintenance programs (which we try to avoid), the data base is updated only once during a transaction.

When a data base change is made, the screen is cleared and a confirming message is displayed. If there were any errors, the data base is not affected, the error message is displayed, and the user has the option of cancelling the entire transaction or correcting the error.

The various program categories described above also make it clear to the user whether the requested program has the capacity to change the data base (data maintenance and processing) or not (inquiry and reporting). For all batch programs and deletes the programs always ask the user to confirm the request by pressing a function key.

PANIC BUTTON: Every system should have a panic button. People laugh when it is described but they certainly feel better for having one. Our panic button is a function key. Whenever it is pressed, and it is always active, the current action, whatever it is, is cancelled and the current program is reinitialized back to what it was when first entered. The data base is unaffected.

This gives users the knowledge that if they do get confused, they can use this key to get them back to a known point so they can start again cleanly.

CONTROLLING THE SYSTEM: It is important that users feel that they are controlling the system, not being controlled by the system. Actions that are required within a specific time-span (such as the logon) tend to be preceived as the system driving the user. Discourteous or abrupt messages communicate the same feeling.

The system should be perceived as supporting the user, guiding where necessary, informing where there is a question, and generally being responsive to the user's desires. Any inflexibility to which the user must conform is a potential problem area. If there are no options in a procedure, then the procedure should be performed by the program without requiring user input. Whenever the user has a input it should be to communicate information or give permission to proceed. Whenever the system takes some action, it should inform the user. For example, if there is a time-out on a function, the system should display that the time limit was exceeded when it aborts the function.

5. PARAMETERIZED AND TABLE-DRIVEN:

Actual numbers are minimized within the coding as much as possible. This permits the user to set the actual values, making the system very flexible. Even where user specification of the parameter does not make sense currently, or is too complex, we still implement it in that manner to allow for future enhancement.

Of course this may have undesirable performance implications. However, it is easier to improve specific performance problems than to generalize a specialized program.

When a rule is implemented within the system, we often put a flag on it, just in case that rule may not be always desired.

The design attempts to have entities defined by a set of codes and values and reduce any complex inter-field relationships. Even if only one method is currently specified, we make it specified by a parameter, just in case. In other words, we rarely hard code a method into an algorithm without having an easy method for specifying an alternative method.

To the extent possible we use the facilities of the HP 3000 to make a system as customizable as possible and yet minimize program changes. For a few examples:

- Every user may have their own menu tree
- There may be any number of data bases running off the same set of programs.
- Custom programs may be integrated into the menus so they are indistinguishable from standard programs.
- Menu trees may be completely customized
- All messages may be customized
- All data base passwords are in an external file, and thus password changes do not require program recompiles.
- What components of the standard product are available at a particular installation are specified by a set of flags accessible to all programs.
- The system as a whole, and each component, has a "Control File" that contains the "configuration", defaults, and parameters for the system and that component. These may be changed without affecting the program code.
- A Code Table provides the translation between the external codes used in a particular installation (which are user specified) and the internal codes (which are fixed by the implementation).

6. RELIABILITY

Reliability is related to consistency, except that it is undesirable that a system be consistently unreliable. Reliability means that the system does what the user expects it to do. Thus it is a function of both the system implementation and the users expectations.

All we have said above can be summarized in this quality. This goal must be part of the system from the very beginning, guiding the design, the implementation, and the training.

V. CONCLUSIONS AND SUMMARY

The large number of issues addressed in this paper, some of them very attitude-oriented rather than specific rules, can be summarized in the following conclusions:

1. User-Friendly Systems are more successful from the perspective of all involved with the system.

2. User-friendly systems can cost the same or less than systems built without this consideration.

3. The considerations that lead to user-friendly systems are consistent with the considerations of the software development team of cost, timing, efficiency, etc.

USER FRIENDLY IS GREAT! BUT WHAT ABOUT US OPERATORS?

Chris Sabean and Jim Dowling

Bose Corporation
Framingham, Massachusetts

ABSTRACT

An overview of the Bose/HP3000 environment and System Operation Procedures are presented. Such aspects as Application System Administration, Batch Scheduling, Tape and Print Handling, System Backup and Recovery, Hardware Maintenance, Fault Isolation and Control, System Performance and Activity Reporting and maintaining a good sense of humor through it all are discussed. The Presentation of this paper will be done by the Authors and some friends in an interactive forum for new HP3000 users.

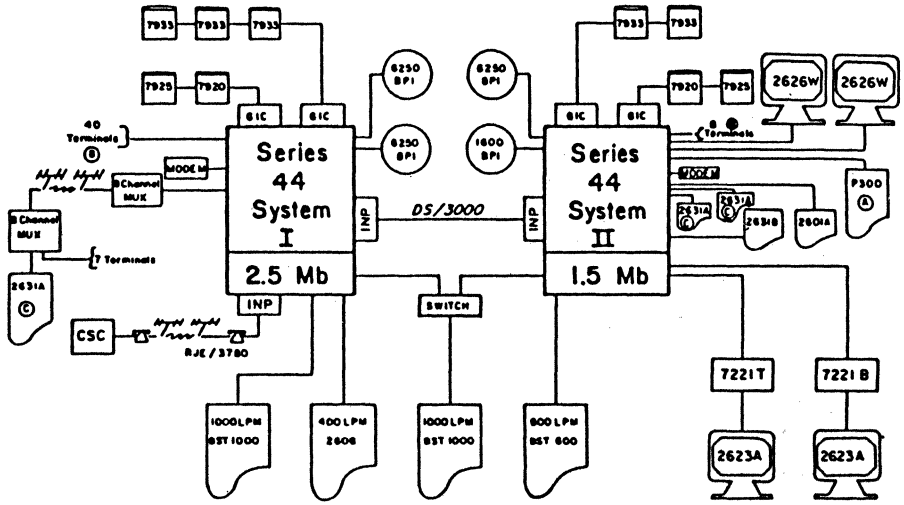
CONTENTS

INTRODUCTION

Bose Corporation is a Manufacturer of Loudspeakers and Audio Electronics, with manufacturing subsidiaries located in Canada and Ireland. We are a World leader in the Consumer HI FI, Professional Sound Systems and Car Stereo market place with Sales subsidiaries located in the United States, Europe, Far East and Australia, with sales in excess of $100 million/year.

We represent Corporate Data Processing services and currently service two manufacturing plants; an Electronics Division is located in Hopkinton and our Framingham plant houses the company's Loudspeaker Division.

Our current configuration of HP3000's is diagrammed below:



NOTES: A). Printronix P300 with TAC board for production of Bar Coded Labels.

B). Terminal Types:

HP2640B, HP2645A, HP2649A, HP2621P, HP2624B, HP2382A

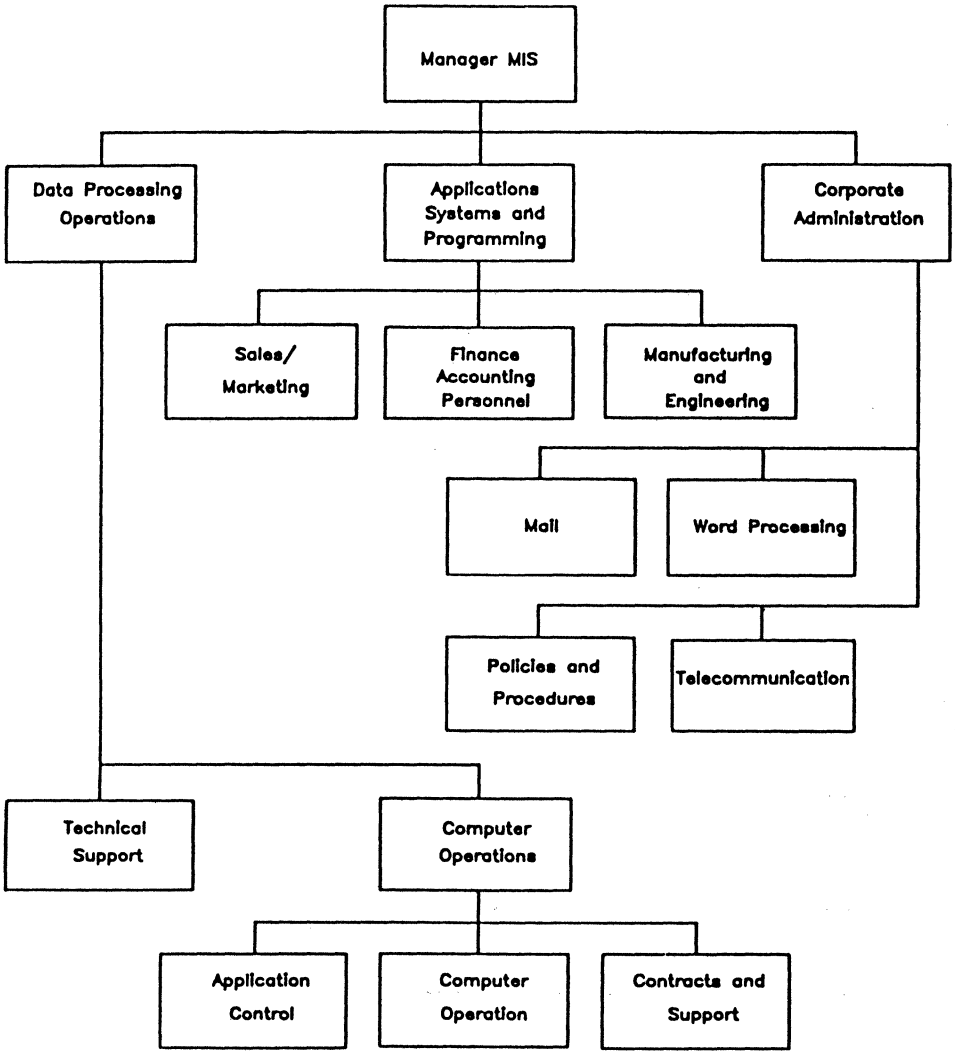
DIRECT 825, 1025

C). HP2631A used as remote printers for Purchasing and Receiving.

D). Use of our ROLM CBX for port contention is imminent.

E). Use of HP Cluster Controller to replace MUX is imminent.

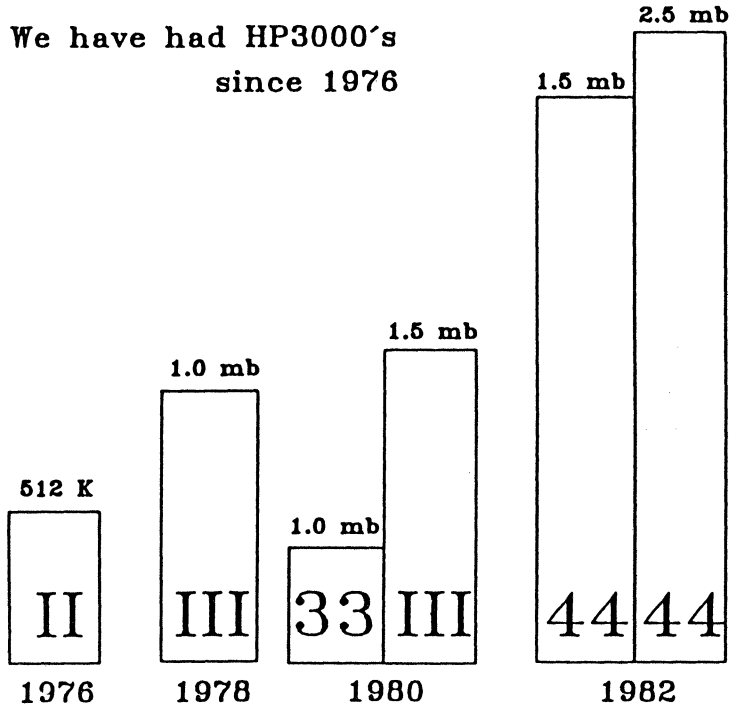
Corporate Data Processing Organizational Chart



ACCOUNTING STRUCTURE DIAGRAM

MPE OPERATING SYSTEM

USERS	ACCOUNTS	GROUPS
MANAGER	SYS	BOSE PRIVATE PUB
FIELD	SUPPORT	HP32080 PUB
MGR	TECH	JOB PUB SOURCE
MGR OPER ORDER STOCK	AAPROD	SRCLIB STREAMS TEXTFILE USEFILE OPI MITS PUB
MGR	SIC	IMAGE PUB
MGR	OTHER...	PUB...



SCRIPT

This paper will address the Computer and Production Environment. The way we will present it is through a "Day in the Life" Scenario. We wish it were Fiction. The data is intended as food for thought. We hope that some of our tools, and procedures will be of value to you in your D/P Environment.

DAY 1

COMPUTER OPERATIONS

18:00 Log-on :HELLO CS3280,OPER.AAPROD,STREAMS
Shut Down On-Line Access for Batch Processing (PUDC)

18:15 Manually STREAM SESDBKUP (this will subsequently stream SESD014)
Log start time and Job # on Production Schedule

18:20 SESD014 aborts - Failure is logged
Unusual Condition Report (UCR) is generated
Production Run Sheet is checked and recovery is performed
Recovery job (SESRRSTR) is executed and it aborts
Call Divisional System Manager (DSM)
Establish recovery - with 2 subsequent jobs
Run time - 45 minutes
Print time - 10 minutes
Decision - Hold until next morning
Generate Report of Reports for each Job that will not run
Generate & distribute Missing Report Form (MRF) to Users
Log subsequent jobs that will be delayed on Production schedule

18:30 Continue with Nightly Production

18:50 Datedump
2 Fulldumps
Today is RELOAD day

21:00 SHUTDOWN
RELOAD ACCOUNTS
FILE DUMPTAPE;DEV=TAPE
FILE SYSLIST;DEV=LP
RESTORE *DUMPTAPE;@.@.@;OLDDATE;FILES=12000;SHOW

22:00 Verify SYSLIST, generate UCR for files not restored, then file listing
Log Downtime
Log Tapes
Package tapes for offsite storage - 1 tape set remains in-house for possible file retrieval
Lockout BARCODE.AAPROD and SERIALS.AAPROD (PUDC) to prevent access to system until the problem (Above) is fixed
Log on as OPERATOR.SYS
Log on UDC raises limits, enables logging, etc....
Go home

DAY 2

COMPUTER OPERATIONS

7:00 Check previous days Production Schedule for ABORTS, UCR's, etc.
Review all UCR's
Set up days production stream files by installing run time parameters via USE FILES
Ensure SLEEPER is running
Issue System Problem Report (SPR) and note what Users have been locked out, then deliver to Application Control Group for processing

8:00 Coffee Break

8:14 Operator is Paged

User: Is the system up?
Operator: It was 10 minutes ago.
User: Then my television is hung.
Operator: I'll check right away and call you back.

8:20 At system console: SYSTEM FAILURE #650
STATUS %1001100 DELTA P 012542

Log System down time
Write UCR
Log in Gold Book
Check Production Schedule for Jobs running at time of failure and record on Production Schedule.
Look up SF #650 in Console Operators Manual
Recovery is: MEMORY DUMP
WARMSTART until INITIAL detects defective tracks

Look up Recovery in BOSE Operators Manual
Perform: MEMORY DUMP
WARMSTART - Note: No defective tracks listed
SPOOK OUTPUT SPOOLFILES (Use ;PURGE option)
SHUTDOWN
COOLSTART
Note: OPERATOR.SYS logon UDC
Send (BROADCAST) message to users
Write UCR on Warmstart
Issue SPR on Warmstart & Deliver to Application Control Group
Issue SPR on System Failure #650
Log System Up time
Catalog Dump Tape
Stream DUMPJOB
Run SPOOK (Spookin spoolfiles)
Deliver DPAN% list (Memory Dump) to System Manager
Check output from Jobs running at time of System Failure and log on Production Schedule and schedule recovery and/or rerun.
Issue UCR or SPR

DAY 2 (Cont'd)

<< SWICH SCENE TO APPLICATION CONTROL GROUP >>

08:00 Review SPR, enter into system and distribute to DSM and file appropriate copies

Review recent Change Action Sheets that effect SESD014 or any programs used in it

<< SWITCH SCENE TO DSM >>

8:30 Review of last nights SPR's

SESD014 $STDLIST shows:
Program SE014P was attempting to read file SE014IN and OPEN Service was not granted
Tombstone shows FSERR 47 (DISCIO ERROR ON FILE LABEL) on LDEV 15.

Recovery Job SESRRS1R $STDLIST shows:
Restore of file SE014IN failed due to FSERR 38 (parity error on tape)
Manual Recovery effected
Issue Operations Processing Request (OPR) to:
Unlock BARCODE.AAPROD and SERIALS.AAPROD (PUDC)
Rerun SESD014 and subsequent jobs at noon
Close SPR

<< SWITCH SCENE TO SYSTEM MANAGER >>

8:30 Review last nights SPR's.

SF 650 - Suspect Power Disturbance!
Review Gold Book and recent SPR's for additional information
No obvious related Unusual Conditions noted

Warmstart - System Manager Manual & Console Operators Manual indicates that a Suspect Tracks Warning should be issued.

Call PICS - SE indicates probable Hardware defect. Memory dump is no good

Call CEO - CE indicates that since no Drive Fault condition existed we must wait till next occurance or give up the machine for Diagnostics. Memory dump is no good. Probably caused by Power Disturbance.

Hold both SPR's open to see if symptom recurs

<< SWITCH SCENE BACK TO COMPUTER OPERATOR >>

8:30 All seems normal through the day

DAY 3

07:00 Everything is normal - no UCR's from last night

08:00 Coffee Break

08:45 Back at Console
SYSTEM FAILURE #650
STATUS %1001100 DELTA P 012542

Same Recovery and results as yesterday
UCR's & SPR's issued for SF 650 and WARMSTART

<< SWITCH SCENE TO SYSTEM MANAGER >>

Readers Digest version of next few days

CE's suggests running **WORKOUT** to cause failure. Workout runs continuously in **DS QUEUE** for days 3, 4 and 5.

System runs fine through day 7.
Must have been Power??

<< BACK TO CONSOLE >>

DAY 8

08:20 SYSTEM FAILURE #650
STATUS %1001100 DELTA P 012542

09:00 Same Recovery & Results as before
UP AND RUNNING!!

09:10 SYSTEM FAILURE #650
STATUS %1001100 DELTA P 012542

09:45 Same Recovery & Results as before
UP AND RUNNING!!!

<< BACK TO SYSTEM MANAGER >>

08:30 Must do something!
Run DPAN4 on most recent Memory Dump
Notice a DISCIO completion failure logged but most other tables are garbaged
DPAN4 another Memory Dump - Same observation.

09:30 Call PICS
Describe the observations then await call back

10:00 Mail brings a new System Status Bulletin (SSB)
Review of MPE reveals the critical clue:
If a DISCIO error occurs while transferring to or from VIRTUAL Memory the system will Fail before the Suspect Track is recorded.
Decoding the Disc Request Table entry indicates the Cylinder and Head on LDEV 15 where the error occurs.
A look at Virtual Memory Allocations indicates that the location is in Virtual Memory.

Schedule further investigation for evening.

10:40 PICS Callback with no new data.

Describe Observations to SE and confirm that it is a likely conclusion.

<< BACK TO OPERATOR >>

21:00 Production is complete
System all backed up

<< BACK TO SYSTEM MANAGER >>

23:00 Deallocate Virtual Memory from LDEV 15.
RUN WORKOUT

24:00 WORKOUT still going ok

Reading through System Manager Manual notice the next clue.
Although VM allocations may be configured on a COOLSTART; the changes will only take effect on a RELOAD.

DAY 9

03:00 RELOAD complete
RUN WORKOUT
SF #650
WARMSTART
Delete Defective Tracks
COOLSTART
RUN WORKOUT
.
.
.
.

It is now clear that the Coolstart that was done three weeks ago to increase Virtual Memory had actually set a Time Bomb which was triggered first by that Production Job Abort but turned lethal when the RELOAD put the piece of Disc into Virtual Memory.

Reallocate Virtual Memory
Make changes on Coolstart
SYSDUMP and create:
Carriage Return Dump
Future Date Dump
Schedule Reload for weekend

... And they all lived happily ever after - (Until the next time)!

CONCLUSION

Our computer facility is a critical corporate resource serving production on-line and batch systems as well as system development. It provides decision support information to all levels and all departments. As such we must provide a reliable, productive and responsive environment to our consumers. These consumers range form data entry clerks through professional programming personnel as well as personal computing users at administrative and executive levels. We have found that the key to our success lies in our ability to:

A) Install new application systems quickly and to operate them as they were designed as consistently as possible.

B) To detect symptoms of problems as soon as possible and make them known to all concerned.

C) Have the capabilities of symptom detection and recovery designed into both manual and automated systems.

D) Ensure that, through documentation, all parties are aware of what is expected of whom and when.

To these ends we have assimilated data from many sources, tried many different techniques and finally constructed the set of Operation Policies and supporting procedures that have been presented here.

The discipline to create and execute procedures down to the lowest level, Log all activity and report unusual conditions has been difficult to develop and maintain. The value of the effort comes when the information necessary to solve a problem is available, when the data necessary for recovery is ready to go, when the review of activities and problems shows a trend that portends future problems that can be dealt with early, and most of all when the confusion, job dissatisfaction and finger pointing that so often "goes with the territory" becomes non-existent.

As you venture into this poorly charted sea of ever changing currents and climate take heart; After seven years of searching, we found most of what we need from those Hewlett Packard manuals; We maintain a good sense of humor; and our users are "Friendly" or else we'll write a policy to make them friendly.

GLOSSARY

Application Control Group - Responsible for: Data Base Administration, Application Installation and Program Maintenance Control

Bose Operators Manual - Bose has created its own set of System Operations Procedures which "Interpret" and enhance the HP manuals.

Broadcast - We use the Contributed Software Library Program "BROADCAST" (modified) to send a message to each hardwired connected terminal to announce that the system is back up.

Carriage Return Dump - A Sysdump (See Sysdump) with a dump date response of (Carriage Return). Dumps MPE and Tables and I/O configuration but no Directory of files.

CE (Customer Engineer) - This is your connection to resolving Hardware Problems. Thou shalt Honor and Respect but accept no Bull from this person.

CEO (Customer Engineer Organization) - This is the TEAM that will keep your hardware running. REMEMBER: Problem escalation is YOUR problem.

Change Action Sheet - Is used to document changes to Production Systems, and inform the requestor of actions taken. (See Exhibit M).

Computer Operations - Responsible for Production Scheduling and machine operation.

Console Operators Manual - HP provides it. You read it. System failure error messages are found in this volume.

COOLSTART - Loads MPE from DISC into Memory from system disc; all permanent user files are saved, but operational environment present prior to last Shutdown is not retained. Open SPOOFLES (Input and Output) go to LOST DISC space. Note: There is no way to detect Lost Disc Space (directly).

Datedump - Stores all files that have been modified since the date pro-

Divisional Systems Manager - Application Systems & Programming Manager for FIN/ACTG, MFG/ENG & SALES. Responsible for all aspects of System Design, Programming and Maintenance. Does not have direct access to "Released" Production Systems.

DPAN4 - Program that produces an analytical formatted listing of main memory contents based on a memory dump taken after system termination. Study of this listing can determine cause of failure, migraine headaches or both. Note: DPAN4 writes to this tape so it needs a write ring.

DS QUEUE - See System Manager/System Supervisor Reference Manual. User processes (Programs) are scheduled for Processor activity in one of three circular prioritized Queues. Typical scheduling is Online priority, CS; Batch, DS which is slightly overlapping CS and Background, LS which is lower than DS.

DUMPJOB - Provided by HP but modified by BOSE. This Job stored in PUB.SYS will produce a Formatted listing of Memory by executing DPAN4 (See DPAN4). It also prints the current I/O Configuration and System Code Load Map.

Files not Restored - This can happen if the Creator, Group or Account has been purged, bad dump tape, out of disc space, etc.

Fulldump - A Sysdump (See Sysdump) with a response to Dump Date of Washington's Birthdate. This dumps MPE, Files, TABLES, I/O and Directory (the whole show). NOTE: Date 0 will write bad file labels and data to tape with a warning.

Future Date Dump - A Sysdump (See Sysdump) with a response to Dump Date of 12/31/99. This dumps MPE, TABLES, I/O and Directory but no files.

Gold book - If if ain't logged here it ain't so! This book comes with your machine at installation time. It should be kept up-to-date and a copy of your system configuration should reside here.

INITIAL - This is the program that gets the machine running at Startup. See System Manager/Supervisor Reference Manual.

LOG Downtime - System Up/Downtime Summary where Date, Time UP/DOWN, Reason System is Down, and Production Schedule delays are logged for weekly reporting purposes. (See Exhibit J).

Log-on HELLO CS3280,OPER.AAPROD,STREAMS - (CS3280) is a BOSE standard where employee types in his Initials followed by his employee number, followed by a comma. This is tied in with our HP3000 Users Distribution List for security, and memo distributions.

Log Tapes - (Operations PRocessing Data Base) Home grown tape Library System showing Tape #, Tape Label Description, Date created, Expiration Date, Tape # of #, Owner and Comments.

Memory Dump - An HP stand alone utility that takes the contents of memory at the time of the system failure and dumps it onto tape. This can only be performed when the system is in a halted state. Discs need not be running but there must be power to the system. (See DPAN4).

Missing Report Form - (MRF) Informs Report Recipients that a scheduled report was not generated and the reson(s) why, whether or not it is rescheduled and who to contact if there are questions. (See Exhibit I).

Offsite Storage - A secured facility located 30 miles away that we contract to store all Archival tapes and Fulldump tape sets on a yearly rotating basis.

Operation Processing Request - To provide a mechanism for requesting Data Processing Operations services or changes to standard operating procedures. (See Exhibit O).

PICS - If you have CSS Software Support you can call for free SE assistance on problems.

Production Schedule - This is the Operators BIBLE, generated daily for a listing of jobs that are scheduled for a particular shift, whether it will be streamed by SLEEPER (see SLEEPER), manually or automatically-by another job, scheduled time for it to run and print, any prerequisite and/or subsequent jobs, the Actual times the job ran, if there are Tapes and special notes. He/she is required to LOG all jobs on this sheet whether on Jobs are on the Schedule or not. (See Exhibit D).

Production Run Sheet - This is the Operators sequel to the BIBLE. It gives all the information he/she needs to get the Job Stream executed and Recovered if necessary. (See Exhibit F).

PUDC - The Contributed Software Library Program has been set up to allow for selective shutdown of Application Systems. (See Exhibit A).

Recovery Job - From Production Run Sheet - A job that is streamed to recover from an abnormal termination of a Production Job. This will get you back to the beginning as if no processing had taken place. (See Exhibit G).

Report of Reports - From Operations PRocessing Data Base a program is inserted at beginning of each Job Stream to produce a report for the Operators control indicating Report Number, Report Name, Mail Stop, Report Recipient, Distribution Station, Number of Copies and Comments. (See Exhibit H).

RELOAD - HPE program that RESTORES your machine contents from a SYSDUMP (see Sysdump) tape. Note: Does not Rebuild Directory or TABLES.

RELOAD ACCOUNTS - Allows you to get your system up even if the 11th reel of the fulldump is bad because no files are restored (as opposed to RELOAD). Configures I/O, HPE and Accounting Structure. Does not restore any files. Note: It does erase the File Directory.

RESTORE *DUMPTAPE;@.@.@;OLDDATE;FILES=12000;SHOW -
Used after RELOAD ACCOUNTS to get files back:

1. OLDDATE for later reference of old create dates.
2. FILES=12000 prevents ABORT due to RESTORE scratch file getting full. Default is 4000.

SE - (Software Engineer) - This person is your connection to Software Problem Solvers. In general he/she will be able to handle all the problems that you were taught to avoid at the System Manager/ Supervisor Course. But, will need help and/or time to resolve less common problems.

SESDBKUP - This is the prerequisite job that is run to STORE the serialization data base and associated files, providing "before IMAGE" backup. (See Exhibit B).

SESD014 - JOB name of current production job using File Naming Standards. (See Exhibit C).
SE - SErialization (Software System designation code)
S - Indicates a STREAM file
D - Run frequency (daily)
014 - User defined 1-3 alphanumeric characters
6th character - Reserved for operations to provide unique names for job which must execute multiple times in one day.

SE014P - Program name using File Naming Standards.
SE - SErialization (Software System designation code)
014 - Program Number (3 digits).
P - Executable program code

SE014IN - File within the Job SESD014 again using File Naming Standards.
SE - SErialization (Software System designation code)
014 - Taken from Program number
IN - Input only file

SHUTDOWN - Shuts down the Operating System in an orderly manner.
Note: WARN @ all Users logged on and manually shut communication lines.

SLEEPER - Contributed Software Library Program which automatically STREAMS and reschedules periodical JOBS.

SPOOK - HP Utility program that allows interogation and operation on spooled devicefiles (SPOOLFILES) created and maintained by MPE. (See System Utilities Reference Manual).

SYSDUMP - MPE program used to Backup and Alter the System Configuration. See System Manager/Supervisor Reference Manual.

SYSLIST - Store and Restore writes the list of files STORED or RESTORED (if ;SHOW is specified) to this file. Usually equated to Printer.
Note: RELOAD will only list on Console Device.

System Problem Report - Indicate error conditions or problems within any functional area of the Information Systems Department which require resolution with regard to any phase in the support of corporate-wide information services. It is a logging, action item, and feedback vehicle to improve information flow, and to insure proper communication and disposition. (See Exhibit L).

System Manager Manual - Provided by HP. Read it.

System Status Bulletin - This is your first line of defense for both Software and some apparent Hardware problems. NOTE: You must determine what Software Versions your are running.

Tombstone - See Exhibit N.

UDC - Command built from a collection of standard MPE commands, and/or other UDC's (see MPE Commands Reference Manual). Can greatly reduce the complexity of such cumbersome tasks as ALTSPOOLFILE #Onnn;PRI=10; DEV=6;COPIES=3 to (PRINT 561,10,6,3) and at the same time increase your Logon nap time waiting for a ":" prompt. (See Exhibit K).

Unusual Condition Report - (UCR) A formal record of all unusual conditions (Sometimes "IT WORKED" is considered unusual) and is used to document symptoms of problems and assure that proper maintenance is performed. (See Exhibit E).

USE FILES - A file containing EDIT/3000 commands. EDIT/3000 reads all commands from the USE file. Any messages from EDIT/3000 are sent to the OUTPUT file, as are requests for text records. Text records then are entered through the INPUT file. See EDIT/3000 Reference Manual. We use USE FILES to install Run Time parameters such as DATES, into Job Streams.

Virtual Memory Allocations - See System Manager/Supervisor Reference Manual.

WARMSTART - Cold loads system from System disc. Will recover incompletely processed spooled jobs and spooled files. Usually the recovered SPOOFLE is destroyed in the process.

WORKOUT - Provided by HP with your new machine (System Verification Tape). It exercises DISC and Tape as well as Memory and CPU.

EXHIBIT A

```
                        BARCODE MENU                                    810
                  WED, FEB 16, 1983, 12:36 PM


     1    OP845P                  (Barcode Label Print Program)
     2    SHOWME                  (Used to Display WELCOME messages)
     3    EDITOR-OP845I01         (BASICENTRY - Single Serial Numbers)

          ENTER  "E"  TO LOG OFF THE SYSTEM

     SELECTION? 2
```

```
@EDITSCR.SRCLIB.AAPROD      WED, FEB 16, 1983, 12:29 PM

 1      C                 <<MENU861 PURCH MAIN MENU>>
 2      :SETJCW MENUC01=0K
 3      C                 <<MENU862 PURCH FRAMINGHAM MENU>>
 4      :SETJCW MENUC02=0K
 5      C                 <<MENU:63 PURCH HOPKINTON MENU>>
 6      :SETJCW MENUC03=0K
 7      C                 <<MENU:64 RECVING MAIN MENU>>
 8      :SETJCW MENUC04=0K
 9      C                 <<MENUC65 RECVING FRAMINGHAM MENU>>
10      :SETJCW MENUC65=0K
11      C                 <<MENU:C6 RECVING HOPKINTON MENU>>
12      :SETJCW MENU:C6=0K
13      C                 <<MENU:16 BARCODE - SINGLE MENU>>
14      :SETJCW MENUC1L=0K
15      C                 <<MENU:17 ORDER MAIN MENU>>
16      :SETJCW MENU:17=0K
17      C                 <<MENU:18 ORDER FRAMINGHAM MENU>>
18      :SETJCW MENUC18=0K
19      C                 <<MENU119 ORDER HOPKINTON MENU>>
20      :SETJCW MENU819=0K
21      C                 <<MENU:30 SERIALS - SEC23P.MITS>>
22      :SETJCW MENU63C=0K
23      C                 <<MENU:31 COOPCRED -  IN852P.OPI>>
```

EXHIBIT B

PRODUCTION RUN SHEET

SYSTEM: Serialization DSN: Mfg./Eng.

JOB NAME: SESDBKUP FREQ: Daily DATE ISSUED: 11/17/86

PREREQ: None RUN SYS: I DATE REVISED: 12/15/86

SUPSEQ: SESDC14 TEMPFILE: NONE USEFILE: NONE

REPORT(S)/DESCRIPTION:
NONE

JOB DESCRIPTION:

Creates a tape of key serialization files to use for error recovery from any Serialization job stream.

Stores: FGDB
SERR
OLDITMST
OLDOHMST
OLDSIMST
SE:290TA
SRLOC
SE:141N
SE:23031 to the SEBKUP tape series.

RUN INSTRUCTIONS:

LOG ON: :HELLO OPER.AAPROD,STREAMS

MPE COMMANDS: :STREAM SESDBKUP

RERUN INSTRUCTIONS:

Same as Run Instructions

SPECIAL RUN INSTRUCTIONS:

LOG ON: :HELLO OPER.AAPROD,STREAMS

MPE COMMANDS: :EDITOR
/T SESDBKUP
/DO - (delete the line that streams SESDC14)
/K GO
/E
:STREAM GO

This will keep the job from stream streaming SESDC14 when it successfully completes.

SECURITY: Standard security. No extra passwords required.

ERROR RECOVERY: Correct problem and re-run SESDBKUP.

EXHIBIT B
(continued)

```
!JOB SESDBKUP.OPER/     .AAPROD.HITS/
!COMMENT ----------------------------------------
!COMMENT      PRINT REPORT OF REPORTS PRINTED
!COMMENT ----------------------------------------
!FILE OPE66BTA=SIPARTIDEV=LPICCTL
!CONTINUE
*RUN  OPL66P.PUB.COMPUTE
$ESDBKUP
!COMMENT ##########################################
!COMMENT   DAILY BACKUP OF KEY SERIALIZATION FILES
!COMMENT   THIS TAPE IS TO BE USED IF A RESTORE IS NEEDED
!COMMENT ##########################################
!COMMENT
!COMMENT ----------------- OPB85P ------------------
!COMMENT -                                         -
!COMMENT - OPB85P IS A TAPE ACCESS PROGRAM.        -
!COMMENT -    IT PROVIDES THE FOLLOWING FUNCTIONS  -
!COMMENT -    FOR SETS OF TAPES WHICH ARE TO BE    -
!COMMENT -    ACCESSED IN CYCLES.                  -
!COMMENT -    I  INPUT-READ ACCESS TO THE LAST     -
!COMMENT -               TAPE WRITTEN TO.          -
!COMMENT -    O  OUTPUT-WRITE ACCESS TO THE NEXT   -
!COMMENT -               TAPE IN THE CYCLE.        -
!COMMENT -    U  UPDATE-WRITE ACCESS TO THE LAST   -
!COMMENT -               TAPE WRITTEN TO.          -
!COMMENT -    R  ROLLBACK-WRITE ACCESS TO THE      -
!COMMENT -               PREVIOUS TAPE IN THE CYCLE-
!COMMENT -                                         -
!COMMENT - PARM CARD #1                           -
!COMMENT -  CC  1- 8 SYSTEM NAME                  -
!COMMENT -       9-10 FILLER                      -
!COMMENT -      11-48 OPTIONAL CONSOLE MSG.       -
!COMMENT -        49    'A' OPTIONAL SECOND CARD   -
!COMMENT -                  AS WO OPTIONAL SECOND CARD.-
!COMMENT -        50    I= INPUT, O=OUTPUT         -
!COMMENT -                  U=UPDATE, R=ROLLBACK   -
!COMMENT -        51    S=SOFT FAIL ON ERRORS.     -
!COMMENT -                  H=HARD FAIL ON ERRORS. -
!COMMENT -      52-59 FILENAME OF CYCLE FILE.      -
!COMMENT -      60-67 GROUP    OF CYCLE FILE.      -
!COMMENT -      68-75 ACCOUNT  OF CYCLE FILE.      -
!COMMENT -      76-80 FILLER (REQUIRED!!!).        -
!COMMENT -                                         -
!COMMENT - PARM CARD #2                           -
!COMMENT -  CC  1-80 FREE FORM. THIS CARD IS       -
!COMMENT -           ISSUED AS AN MPE COMMAND     -
!COMMENT -           (COMMAND INTRINSIC).         -
!COMMENT -           IF THE STRING '2222' IS ON   -
!COMMENT -           THE CARD THE TAPE NUMBER     -
!COMMENT -           WILL BE INSERTED BEFORE ISSUE.-
!COMMENT -                                         -
!COMMENT -  SEE OPB85P RUN SHEET FOR MORE INFO.    -
!COMMENT -  SEE ALSO OPCCSP.DOC.UTIL.              -
!COMMENT -----------------------------------------
!COMMENT
!RUN OPCCSP.PUB.UTIL
$ESKUP                                  TOMOPCBICYCPUB     AAPROD

FILE SESTORE=DT2222@IDEV=TAPEINDDUF
!COMMENT ------------------------------------------------------------
!COMMENT - *OP323P.PUB.UTIL WILL PROMPT FOR A DATA BASE NAME.      -
!COMMENT - THEN ATTEMPT TO *DBOPEN* THAT DATA BASE EXCLUSIVELY.    -
!COMMENT - IF IT IS SUCCESSFUL A REQUEST TO RELEASE IT WILL BE     -
!COMMENT - ISSUED TO THE MASTER CONSOLE FOR PERMISSION TO          -
!COMMENT - TO RELEASE IT. IF THE ANSWER IS NOT 'Y' THE REQUEST     -
!COMMENT - WILL BE REISSUED REPEATEDLY UNTIL PERMISSION IS         -
!COMMENT - GRANTED. IF THE DATA BASE IS ALREADY BEING ACCESSED     -
!COMMENT - A MESSAGE WILL BE ISSUED TO THE MASTER CONSOLE NOTING-  -
!COMMENT - THE 'BUSY' CONDITION. AT THAT POINT THE PROGRAM WILL    -
!COMMENT - PAUSE FOR 33 SECONDS THEN MAKE ANOTHER ATTEMPT AT THE-  -
!COMMENT - EXCLUSIVE *DBOPEN*. TWENTY SUCH ATTEMPTS WILL BE MADE-  -
!COMMENT - BEFORE THE PROGRAM WILL END. IN THE CASE OF AN          -
!COMMENT - INABILITY TO OBTAIN EXCLUSIVE ACCESS TO THE DATA BASE-  -
!COMMENT - THE PROGRAM WILL SET THE JOB CONTROL WORD *JCW* TO      -
!COMMENT - '999' THEN TERMINATES IF THE DATA BASE IS ACCESSIBLE -  -
!COMMENT - *JCW* WILL BE SET TO -8-1 OTHERWISE *JCW* WILL BE       -
!COMMENT - SET EQUAL TO THE *DBOPEN* RETURN CODE.                  -
!COMMENT ------------------------------------------------------------
!COMMENT
!COMMENT ------------------------------------------------------------
!COMMENT      OBTAIN EXCLUSIVE ACCESS OF PGDB DATA BASE
!COMMENT ------------------------------------------------------------
!RUN OP323P.PUB.UTIL
PGDB
!COMMENT
!COMMENT ------------------------------------------------------------
!COMMENT      OBTAIN EXCLUSIVE ACCESS OF SCRA DATA BASE
!COMMENT ------------------------------------------------------------
!RUN OP323P.PUB.UTIL
SCRA
!COMMENT
!COMMENT ------------------------------------------------------------
!COMMENT   PGDB DATA BASE, SCRA DATA BASE, AND KEY SEQUENTIAL
!COMMENT   FILES WILL NOW BE BACKED UP
!COMMENT ------------------------------------------------------------
!COMMENT
!STORE PGDB@.SCRA@.@PLOC.@LOGTMST.@LOGMMST,@LOGSMST,SESB290TA.6
!SECOND=SESB230811=SESTORE!SHOW
!COMMENT
!STREAM SESDBK14.STREAMS
!TELLOP! ********** ********** ****************
!TELLOP!   SESDBKUP HAS NOW RUN TO EOJ
```

EXHIBIT C

PRODUCTION RUN SHEET

SYSTEM: Serialization DSM: Mfg./Eng.

JOB NAME: SESD014 FREQ: Daily DATE ISSUED: 12/31/79

PREREQ: SESDBKUP RUN SYS: 1 DATE REVISED: 12/15/82

SUBSEQ: SESDC1@ TEXTFILE: None USEFILE: None

REPORT(S)/DESCRIPTION:

1. SED14RC1,Serialization Error Report

REMARKS/COMMENTS/SPECIAL INSTRUCTIONS:

This job is automatically streamed by SESDBKUP.

JOB DESCRIPTION:

This job converts the MSI transmissions into a COBOL-usable file, SE@14OT. It also streams SESD?18.

RUN INSTRUCTIONS:

Automatically streamed by SESDBKUP.

RERUN INSTRUCTIONS:

This job may not be re-run. See instructions for error recovery.

SECURITY: Standard security. No extra passwords required.

ERROR RECOVERY:

A. If system fails:
LOG ON: :HELLO OPER.AAPROD,STREAMS
:STREAM SESRASTR
:STREAM SESD314

B. If job aborts:
1. Notify responsible DSM to fix the problem.
2. Restream job using Error Recovery Instructions "A" above.

EXHIBIT C
(continued)

```
!JOB SESD014,OPER/      .AAPROD,MITS/
!COMMENT ---------------------------------------
!COMMENT      PRINT REPORT OF REPORTS PRINTED
!COMMENT ---------------------------------------
!FILE OPC06OTA=S1PART;DEV=LP;CCTL
!CONTINUE
!RUN  OPSC6P.PUB.COMPUTE
SESD014
!PURGE SEC29RC2
!PURGE SE:230LD
!PURGE SAVS1MST
!PURGE SAVOHMST
!PURGE SAV1TMST
!PURGE SE:140T
!PURGE SE:16ST
!PURGE SE:140LD
!PURGE SE:280TA
!PURGE SEC28STA
!PURGE SE:150TC
!PURGE SE:15ST
!PURGE NEWOHMST
!PURGE NEW1TMST
!PURGE NEWS1MST
!COMMENT ------------------------------------
!COMMENT  THIS PROCESS APPENDS SCREEN INPUT
!COMMENT  TO TAPE TRANSMISSION INPUT.
!COMMENT
!COMMENT  IF SESD:14 IS RERUN ALONE THE FCOPY
!COMMENT  FROM FILE WILL BE EMPTY.  THIS IS OK.
!COMMENT ------------------------------------
!FILE SE:14IN;ACC=APPEND
!RUN FCOPY.PUB.SYS
FROM=SE023001;TO=*SE:14IN
EXIT
!COMMENT ------------------------------------
!COMMENT  SAVE OLD SCREEN INPUT FILE FOR HISTORY
!COMMENT  AND BUILD NEW EMPTY SCREEN INPUT FILE.
!COMMENT ------------------------------------
!RENAME SE023001,SE023OLD
!BUILD SEC230C1;REC=-48,16,F,ASCII;DISC=5000,16,6
!COMMENT ------------------------------------
!COMMENT  EDIT DATA USING SE014P
!COMMENT ------------------------------------
!BUILD SEC140T;REC=-30,17,F,ASCII;DISC=25000,16,8
!BUILD SEC29R02;DEV=DISC;REC=-80,,F,ASCII
!FILE SE014I01=SEC14IN,OLD;ACC=INOUT
!FILE SE:140C1=SEC140T,OLD;ACC=OUT
!FILE SE:14RC1=S1PART;DEV=LP,8;CCTL
!FILE SE014OC2=SEC29R02,OLD
!RUN SE014P
!TELLOP; SERIALIZATION REPORTS ARE NOW STREAMED
!STREAM SESD018.STREAMS
!TELLOP; ********** ********** ********** *****
!TELLOP; * * * SESDC14 HAS RUN TO EOJ * * *
!TELLOP; ********** ********** ********** ******
!RUN OP012P.PUB.AAPROD
SESD:14
```

EXHIBIT D

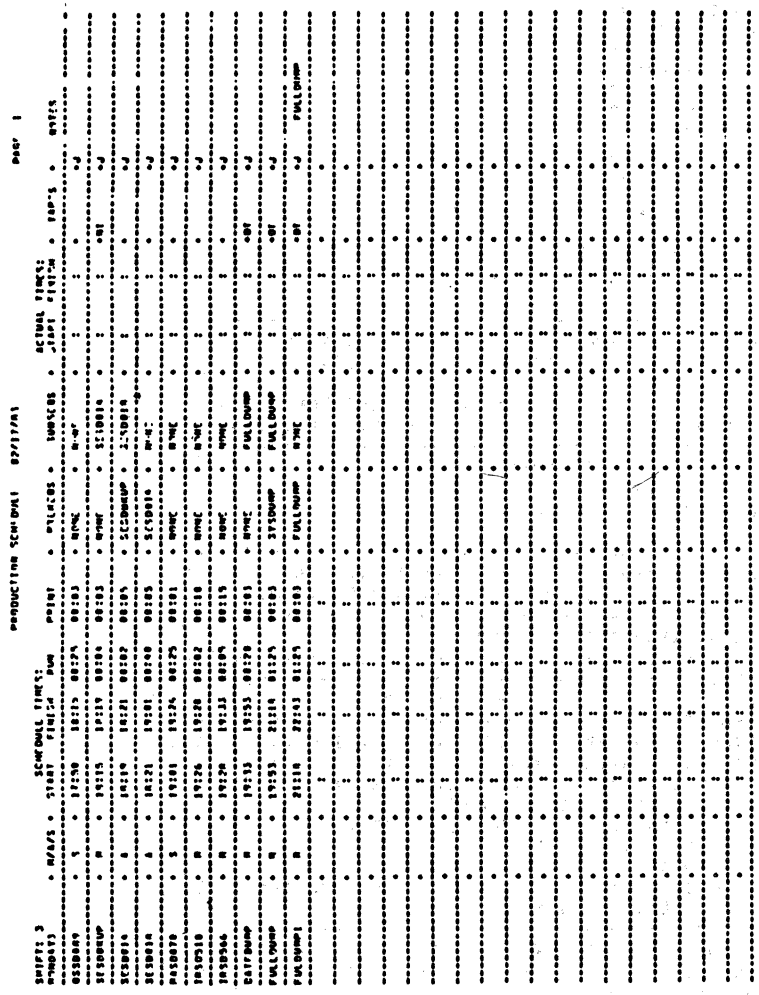


EXHIBIT E

Page 1

REPORT #_____

UNUSUAL CONDITION REPORT

YOUR NAME _____DATE__/__/__TIME _____

SYSTEM_____JOB NAME_____REPORT#_____PROG.NAME_____

USER CONTACT _____ DATE__/__/__AND TIME PROBLEM REPORTED_____

WHAT HAPPENED? _____

_____

_____

_____

_____

_____

_____

TEMPORARY FIX _____

_____

_____

_____ BY WHOM _____

LONG-TERM FIX _____

_____

_____

_____

_____ BY WHOM _____

RECOMMENDATIONS/NOTES _____

_____

_____

_____

_____

EXHIBIT F

PRODUCTION RUN SHEET

SYSTEM: Serialization DSM: Mfg./Eng.

JOB NAME: SESD#18 FREQ: Daily DATE ISSUED: 12/31/79

PREREQ: SESDBKUP RUN SYS: I DATE REVISED: 12/15/82
SESD014

SUBSEQ: None TEXTFILE: None USEFILE: None

REPORT(S)/DESCRIPTION:

1. SE018RC1.Ship Log
2. SE019PC1.Ship Summary
3. SE02C*C1.Ser.Packing List
4. SE021*C1.*fg. Summary
5. SE021RC2.Prg. St. Serial #
6. SE022*C1.Returned Goods Sup.
7. SE022PC2.Returned Goods Listing
8. SE024RC1.Serialization Error Rpt.
9. SE025R21.Sum. Goods Ret'd to Mfg.
10. SE025RC2.Listing Goods Ret'd Mfg.
11. SE029PC1.Inventory Transactions
12. SE029RC2.Error/Exception Report
13. SE031PC1.Inventory Summary

REMARKS/COMMENTS/SPECIAL INSTRUCTIONS:

This job is automatically streamed by SESD:14.

JOB DESCRIPTION:

This job prints out reports based on Finished Goods. It also updates serialization sequential files and the FGDB data base.

RUN INSTRUCTIONS:

Automatically streamed by SESD014.

RERUN INSTRUCTIONS:

This job may not be re-run. See instructions for error recovery.

SECURITY: Standard security. No extra passwords required.

ERROR RECOVERY:

A. If system fails:
LOG ON: :HELLO OPER.AAPROD,STREAMS
:STREAM SESRRSTR
:STREAM SESD014

B. If job aborts:
1. Notify responsible DSM to fix the problem.
2. Restream using Error Recovery Instructions "A" above.

EXHIBIT F
(continued)

```
!JOB SESDC18,OPER/     ,ASPROD,MITS/
!COMMENT ----------------------------------------
!COMMENT      PRINT REPORT OF REPORTS PRINTED
!COMMENT ----------------------------------------
!FILE OPC86OTA=$1PARTIDEV=LPICCTL
!CONTINUE
!RUN  OPC6P.PUB.COMPUTE
$LSD910
!FILE SEC:101I01=SE0140T
!FILE SEC:101C2=SRLOC
!FILE SEC:10RC1=N1PARTIDEV=LPA,4,61CCTL
!RUN  SEC:10P
!FILE SEC:191I1=SE0140T
!FILE SEC:191C2=SRLOC
!FILE SEC:119RC1=N1PARTIDEV=LPA,4,21CCTL
!RUN  SEC:19P
!FILE SEC:2C101=SE0140T
!FILE SEC:201C2=SRLOC
!FILE SEC:20RC1=NCR3PARTIDEV=LPA,4,21CCTL
!RUN  SEC:2CP
!FILE SEC:211N=SE0140T
!FILE SEC:221R21=N1PARTIDEV=LPA,4,41CCTL
!FILE SEC:121R21=N1PARTIDEV=LPA,4,11CCTL
!RUN  SEC:21P1MA=DATA=16E08
!FILE SEC:221N=SEC140T
!FILE SEC:222A21=N1PARTIDEV=LPA,4,11CCTL
!FILE SEC:22RC2=N1PARTIDEV=LPA,41CCTL
!RUN SEC:22P
!FILE SEC:251N=SE0140T
!FILE SEC:25RC1=N1PARTIDEV=LPA,4,11CCTL
!FILE SEC:25RC2=N1PARTIDEV=LPA,4,11CCTL
!RUN SEC:25P
!FILE SEC:24RC1=N1PARTIDEV=LPA,4,21CCTL
!RUN SE324P
!FILE SEC:201NA=SE0140T
!FILE SEC:2ADTAIREC=13,19,F,ASCII!DISC=8000,16,11SAVE
!RUN  SEC:2AP
!FILE INPUT=SE3260TA
!FILE OUTPUT=SEC:20STA
!RUN SORT.PUB.SYS
#KEY 1,18
!END
!FILE SEC:16INA=OLDITMST,OLD
!FILE SEC:16INB=SEC:20STA
!FILE SEC:29A02=SE329A02!ACC=APPEND
!FILE SEC:16OTA=NEWITMST!RECC=13,19,F,ASCII!16
! DISC=35000,16,21SAVE
!FILE SEC:16OTB=SE0140T!ACC=APPEND
!RUN SE31AP
!FILE INPUT=SE0140T
!FILE OUTPUT=SEC:16ST
!RUN SORT.PUB.SYS
#KEY 7,12
!END
!FILE SEC:15INA=OLDOHMST,OLD
!FILE SEC:15INB=SE314ST
!FILE SEC:29RC3!ACC=APPEND
```



```
!FILE SEC:15OTA=NEWOHMST!RECC=17,19,F,ASCII!16
!DISC=20C9C0,16,141SAVE
!FILE SEC:15OTC!RECC=16,8,F,ASCII!DISC=8000,16,11SAVE
!FILE SEC:15OTD=SEC140T!ACC=APPEND
!RUN SE015P
!FILE SEC:17INA=OLDSIMST,OLD
!FILE SEC:17INB=SEC:15OTC
!FILE SEC:17OTA=NEWSIMST!RECC=16,8,F,ASCII!16
! DISC=5P3P0,16,11SAVE
!RUN SEC:17P
!FILE INPUT=SE0140T
!FILE OUTPUT=SEC:15ST
!RUN SORT.PUB.SYS
#KEY 25,2
#KEY 19,6
#KEY 7,12
!END
!FILE SE0291C2=SRLOC,OLD
!FILE SEC:29IC1=SE015ST
!FILE SEC:29R01=S1PARTIDEV=LP,8,T1CCTL
!FILE SEC:29R02!ACC=APPEND
!FILE SEC:29DE1=SE0290TA!ACC=APPEND
!RUN SE029P
!FILE SEPAT=$1PARTIDEV=LP,8,2
!RUN PCOPY.PUB.SYS
FROM=SE029A02!TO=*SEPAT
EXIT
!FILE SE031R01=N1PARTIDEV=LPA,4,81CCTL
!FILE SRLOC,OLD
!RUN  SEC:31P
!RENAME OLDOHMST,SAVOHMST
!RENAME NEWOHMST,OLDOHMST
!RENAME OLDITMST,SAVITMST
!RENAME NEWITMST,OLDITMST
!RENAME OLDSIMST,SAVSIMST
!RENAME NEWSIMST,OLDSIMST
!PURGE SEP140LD,MITS
!RENAME SE0140N,SE0140LD
!BUILD SEC:14INTACC=-48,16,F,ASCII!DISC=10000,16,4
!TELLOP! ***SERIALIZATION UPDATES DONE***
!TELLOP! ** *  SESD816 HAS NOW RUN TO EOJ * * *
!COMMENT ***END OF SERIALIZATION REPORTS***
!RUN OPS12P.PUB
$ESD*18
!EOJ
```

EXHIBIT G

```
                        PRODUCTION RUN SHEET

  SYSTEM: Serialization                                    DSN: Mfg./Eng.
-------------------------------------------------------------------------
JOB NAME: SESRRSTR          FREQ: As Required     DATE ISSUED: 11/17/81

  PREREQ: SESDBKUP       RUN SYS: 3              DATE REVISED: 12/15/82

  SUBSEQ: NONE          TEXTFILE: NONE                USEFILE: NONE
-------------------------------------------------------------------------
REPORT(S)/DESCRIPTION:

1. NONE
-------------------------------------------------------------------------
REMARKS/COMMENTS/SPECIAL INSTRUCTIONS:

          For error recovery only.  Uses tape series SEBKUP.
-------------------------------------------------------------------------
JOB DESCRIPTION:

          Error recovery for any serialization job.

          Restores: FGDB
                    SEPR
                    OLDITMST
                    OLDOHMST
                    OLDSIMST
                    SE?29CTA
                    SRLOC
                    SE:141N
                    SE923C:1      from the SEBKUP tape series.
-------------------------------------------------------------------------
RUN INSTRUCTIONS:

      LOG ON:    :HELLO OPER.AAPROD,STREAMS

MPE COMMANDS:    :STREAM SESRRSTR

-------------------------------------------------------------------------
RERUN INSTRUCTIONS:

      SAME AS RUN INSTRUCTIONS

-------------------------------------------------------------------------
SECURITY: Standard security.  No extra passwords required.

-------------------------------------------------------------------------
ERROR RECOVERY:

          If job aborts, correct problem and restream SESRRSTR
          using Run Instructions.
```

EXHIBIT G
(continued)

```
!JOB SESRRSTR,OPER/      .AAPROD,MITS/
!COMMENT ----------------------------------------
!COMMENT     PRINT REPORT OF REPORTS PRINTED
!COMMENT ----------------------------------------
!FILE OPE660TA=SIPARTIDEV=LP;CCTL
!CONTINUE
!RUN  OP266P.PUB.COMPUTE
SESRRSTR
!COMMENT ************************************************
!COMMENT   RESTORE KEY SERIALIZATION FILES FROM CYCLED TAPES
!COMMENT ************************************************
!COMMENT
!COMMENT ---------------------------------------------
!COMMENT   SET THE STORE TAPE FOR THE RESTORE
!COMMENT ---------------------------------------------
!COMMENT
!COMMENT --------------- OPE05P -----------------
!COMMENT -                                       -
!COMMENT - OP005P IS A TAPE ACCESS PROGRAM.      -
!COMMENT -     IT PROVIDES THE FOLLOWING FUNCTIONS -
!COMMENT -     FOR SETS OF TAPES WHICH ARE TO BE   -
!COMMENT -     ACCESSED IN CYCLES.                 -
!COMMENT -     I  INPUT-READ ACCESS TO THE LAST    -
!COMMENT -              TAPE WRITTEN TO.           -
!COMMENT -     O  OUTPUT-WRITE ACCESS TO THE NEXT  -
!COMMENT -              TAPE IN THE CYCLE.         -
!COMMENT -     U  UPDATE-WRITE ACCESS TO THE LAST  -
!COMMENT -              TAPE WRITTEN TO.           -
!COMMENT -     R  ROLLBACK-WRITE ACCESS TO THE     -
!COMMENT -              PREVIOUS TAPE IN THE CYCLE -
!COMMENT -                                       -
!COMMENT - PARM CARD #1                          -
!COMMENT -  CC  1- 8 SYSTEM NAME                 -
!COMMENT -      9-10 FILLER                      -
!COMMENT -     11-48 OPTIONAL CONSOLE MSG.        -
!COMMENT -     49    Y= OPTIONAL SECOND CARD      -
!COMMENT -           N= NO OPTIONAL SECOND CARD.  -
!COMMENT -     50    I= INPUT, O=OUTPUT           -
!COMMENT -           U=UPDATE, R=ROLLBACK         -
!COMMENT -     51    S=SOFT FAIL ON ERRORS.       -
!COMMENT -           H=HARD FAIL ON ERRORS.       -
!COMMENT -     52-59 FILENAME OF CYCLE FILE.      -
!COMMENT -     60-67 GROUP    OF CYCLE FILE.      -
!COMMENT -     68-75 ACCOUNT  OF CYCLE FILE.      -
!COMMENT -     76-80 FILLER (REQUIRED!!!).        -
!COMMENT -                                       -
!COMMENT - PARM CARD #2                          -
!COMMENT -  CC  1-80 FREE FORM. THIS CARD IS     -
!COMMENT -           ISSUED AS AN MPE COMMAND     -
!COMMENT -           (COMMAND INTRINSIC).         -
!COMMENT -           IF THE STRING "2222" IS ON   -
!COMMENT -           THE CARD THE TAPE NUMBER     -
!COMMENT -           WILL BE INSERTED BEFORE ISSUE.-
!COMMENT -                                       -
!COMMENT -  SEE OP005P RUN SHEET FOR MORE INFO.  -
!COMMENT -  SEE ALSO OP005P.DOC.UTIL.            -
!COMMENT ----------------------------------------
```

PAGE 2

```
!RUN OPE05P.PUB.UTIL
SCERUP                                         VIHOPESICYCPUR     AAPROD
FILE SCRESTOR=BT2222;DEV=TAPE;NOBUF
!COMMENT
!COMMENT ----------------------------------------------------------
!COMMENT  GAIN EXCLUSIVE ACCESS OF THE FGDB DATA BASE AND PURGE IT
!COMMENT ----------------------------------------------------------
!COMMENT
!COMMENT ********************************************************
!COMMENT * "OP020P.PUB.UTIL WILL PROMPT FOR A DATE BASE NAME,    *
!COMMENT * THEN ATTEMPT TO "DBOPEN" THAT DATA BASE EXCLUSIVELY. *
!COMMENT * IF IT IS SUCCESSFUL A REQUEST TO RELEASE IT WILL BE   *
!COMMENT * ISSUED TO THE MASTER CONSOLE FOR PERMISSION TO        *
!COMMENT * TO RELEASE IT. IF THE ANSWER IS NOT "Y" THE REQUEST   *
!COMMENT * WILL BE REISSUED REPEATEDLY UNTIL PERMISSION IS       *
!COMMENT * GRANTED. IF THE DATA BASE IS ALREADY BEING ACCESSED   *
!COMMENT * A MESSAGE WILL BE ISSUED TO THE MASTER CONSOLE NOTING*
!COMMENT * THE "BUSY" CONDITION. AT THAT POINT THE PROGRAM WILL *
!COMMENT * PAUSE FOR 2? SECONDS THEN MAKE ANOTHER ATTEMPT AT THE*
!COMMENT * EXCLUSIVE "DBOPEN". TWENTY SUCH ATTEMPTS WILL BE MADE*
!COMMENT * BEFORE THE PROGRAM WILL END. IN THE CASE OF AN       *
!COMMENT * INABILITY TO OBTAIN EXCLUSIVE ACCESS TO THE DATA BASE*
!COMMENT * THE PROGRAM WILL SET THE JOB CONTROL WORD "JCW" TO    *
!COMMENT * "999" THEN TERMINATE; IF THE DATA BASE IS ACCESSIBLE  *
!COMMENT * "JCW" WILL BE SET TO -1-1 OTHERWISE "JCW" WILL BE     *
!COMMENT * SET EQUAL TO THE "DBOPEN" RETURN CODE.                *
!COMMENT ********************************************************
!RUN OP020P.PUB.UTIL
FGDB
!RUN DBUTIL.PUB.SYS.PURGE
FGDB/MAINT
!COMMENT
!COMMENT ----------------------------------------------------------
!COMMENT  GAIN EXCLUSIVE ACCESS OF THE SERR DATA BASE AND PURGE IT
!COMMENT ----------------------------------------------------------
!COMMENT
!RUN OP020P.PUB.UTIL
SERR
!RUN DBUTIL.PUB.SYS.PURGE
SERR/MAINT
!COMMENT! ---------------- ------------ ---------
!COMMENT   RESTORE THE FGDB DATA BASE, SERR DATA BASE, AND
!COMMENT   KEY SEQUENTIAL FILES
!COMMENT! ----------------- -------------- -----
!RESTORE *SERESTOR;FGDB0,SERR0,SRLOC,OLDITMST,OLDOMMST,OLDSIMST,&
!SE0290TA,SE6340N,SE0230E;SHOW
!TELLOP; *************! ********************* ****
!TELLOP;    SESRRSTR HAS NOW RUN TO EOJ
!!!      ............... ........... ............
```

REPORT NUMBER	REPORT NAME	MAIL STOP	REPORT RECIPIENT	DIST STATION	NUMBER OF COPIES	COMMENTS
SE010001	SERIAL ERRORS LIST		F. BORGES	B	01	

BOSF CORPORATION — REPORTS / DISTRIBUTION — WED. FEB 16, 1983, 12:02 P
REPORT NO. OP001 — JOB NUMBER: SC50010 — JOB NAME: SERIALIZATION — PAGE NUMBER

REPORT NUMBER	REPORT NAME	MAIL STOP	REPORT RECIPIENT	DIST STATION	NUMBER OF COPIES	COMMENTS
SE010001	SHIP LOG		F. BORGES	B	01	
			P. CAMERON	C	01	
			D. DEMARIA	D	01	
			M. KILPATRICK	E	01	
			L. RIVERA	A	01	
			L. SEDELL	S	01	
SE019001	SHIP SUMMARY		F. BORGES	B	01	
			P. CAMERON	C	01	
SE020001	SER PACKING LIST		F. BORGES	B	02	
			S. BOURGEOIS	B	03	
			P. CAMERON	C	01	
SE021001	RFG SUMMARY		F. BORGES	B	01	
			P. CAMERON	C	01	
			J. KNAPP	K	01	
			R. TURREY	T	01	
SE021002	RFG STISERIAL #		P. CAMERON	C	01	
SE024001	SERIAL ERROR RPT		F. BORGES	B	01	
			P. CAMERON	C	01	
SE022001	RETURNED GOODS SUMMARY		P. CAMERON	C	01	
SE022002	RETURNED GOODS LISTING		P. CAMERON	C	01	
SE025001	SUM GOODS BIN RFG		P. CAMERON	C	01	

EXHIBIT H

EXHIBIT I

PAGE 1

MISSING REPORT FORM

DATE_____

REPORT RECIPIENT _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT NUMBER_____ NUMBER OF COPIES _____

REPORT TITLE _____ JCP NO. _____

JOB NAME_____ UCR NUMBER _____

REASON FOR REPORT BEING MISSING:

JCP RAN BUT REPORT NOT GENERATED _____

JOB ABORTED _____ PRINTER JAMMED _____

SYSTEM FAILED _____ JCR DID NOT RUN _____

OTHER _____

COMMENTS _____

_____

_____

_____

_____

_____

_____

DSM _____

RERUN SCHEDULED? YES _____ NO _____ WHEN? _____

NOTE: THIS PROBLEM HAS BEEN REPORTED TO THE DSM NOTED ABOVE.
PLEASE CONTACT ONLY YOUR DSM FOR RESCHEDULING INFORMATION.

OPERATORS NAME _____

EXHIBIT J

PAGE 1

SYSTEM UP TIME/DOWN TIME SUMMARY

DATE	TIME DOWN	TIME UP	REASON SYSTEM DOWN	PRODUCTION SCHEDULE DELAY

EXHIBIT K

```
LOGON
OPTION NOLIST,LOGON
LIMIT 9,0
STREAMS 10
OUTFENCE 7
JOBFENCE 4
SHOWJOB
DSTAT ALL
JOBSECURITY HIGH
JOBPRI CS,DS
SHOWQ
CONTINUE
RUN HOWUP.BOSE
SHOWJCW JCW
IF JCW >= 2 THEN
  DSCONTROL SYSTEM2,OPEN,NOW
  CONTINUE
  LOG OBALOG,RESTART
ELSE
  SHOWOUT
  TELLOP**********************
  TELLOP SPOOK OUTPUT.  DELETE
  TELLOP SPOOLFILES.  SHUTDOWN.
  TELLOP COOLSTART.
  TELLOP **********************
ENDIF
IF JCW > 10 THEN
    ALLOCATE EDITOR.PUB.SYS
    ALLOCATE SORT.PUB.SYS
    ALLOCATE MERGE.PUB.SYS
    ALLOCATE FCOPY.PUB.SYS
    ALLOCATE QUEDIT.PUB.OPELLE
    LIMIT 4,26
    SHOWJOB
    TELLOP  **********************
    TELLOP  THE LIMITS ARE SET.
    TELLOP  DSLINES ARE OPENED.
    TELLOP  LOGGING IS STARTED.
    TELLOP  ***********************************
    TELLOP  TO SEND A MESSAGE TO ALL LOGGED-OFF
    TELLOP  TERMINALS THAT THE SYSTEM IS UP -
    TELLOP  LOG ON AS OPERATOR.TECH AND TYPE
    TELLOP  IN "TELLUSER".
    TELLOP  **********************
ELSE
    TELLOP  **********************
    TELLOP  LIMITS ARE DOWN.  NO USERS
    TELLOP  CAN LOG ON.  YOU MUST RAISE
    TELLOP  THE JOB AND SESSION LIMITS
    TELLOP  TO ALLOW ACCESS TO THE
    TELLOP  SYSTEM.
    TELLOP  ****************************
ENDIF
TELLOP  ****************************
TELLOP  JOB LIMIT MUST BE 1,2 OR 3.
TELLOP  ****************************
```

EXHIBIT L

SYSTEPS PROBLEM REPORT

PR#_____

****

USER CONTACT_____ INITIATOR _____

SYSTEM _____ DATE INITIATED_____

SUB SYSTEM _____ COST CENTER _____

PROBLEM TITLE_____ NUMBER OF PAGES_____

DESCRIPTION OF PROBLEM/ERROR _____

ATTACHMENT _____

****

DATE RECEIVED _____ CHANGE ACTION # _____

PGR ASSIGNED_____ DATE INSTALLED _____

RESPONSIBLE I/S AREA

___ MFG-MANUF./ENG. ___ OPS-- DATA CTR/TECH SERV/DP ADMIN

___ SLS-MKT/SALES/CUST SERV ___ P&P-- POLICIES & PROCEDURES

___ FIN--FINANCE/PERS/ADMIN ___ ADM-- ADMINISTRATIVE SERV.

****

PERSONNEL ASSIGNED _____ACTUAL HOURS _____

ANALYSIS _____

RESOLUTION _____

INITIATOR APPR._____COMPLETION DATE_____NAME_____

EXHIBIT M

CHANGE ACTION SHEET

DATE_____ CHANGE ACTION #_____

NEW___ PRB or SPR#_____

PROG SOURCE_____ ___ B
___ F
___ E
___ $ #J_____
___ M $YS 1]
___ DSCOPY $YS 1
GROUP _____

JOB STREAM _____

COMPILE STREAM_____

OTHER_____

INITIATOR_____ ___ [JOB STRM $YS 1]
___ DSCOPY $YS 1
___ [$YS 1 (PASSWORDS)
GROUP_____

DOCUMENTATION AFFECTED:

	REQ CHG	CHANGED		REQ CHG	CHANGED
PROGRAM SOURCE	_____	_____	OPERATIONS DOC.	_____	_____
PROG SPEC SHEET	_____	_____	PROD. RUN SHEET	_____	_____
FILE SPEC SHEET	_____	_____	JOB STREAM	_____	_____
VIEW SCREEN	_____	_____	DATA BASE	_____	_____
DEL SCREEN	_____	_____	COPYLIB MEMBER	_____	_____
REPORTS/DIST	_____	_____	FILE CHANGE	_____	_____
COMPILE STREAM	_____	_____	DICT/SCHED/TOC	/__/__/__	/__/__/__

DESCRIPTION OF CHANGE(S):

ACCEPTED BY:
PROJECT LEADER_____ DATE_____

OPERATIONS SUPERVISOR_____ DATE_____

____ CHANGE ACTION ACCEPTED AND IMPLEMENTED

____ CHANGE ACTION REJECTED: SEE ATTACHED FOR DETAILS

IMPLEMENTED BY _____ DATE_____

EXHIBIT N

FILE INFORMATION DISPLAY

In addition to Command Interpreter and run-time (abort) error messages, certain file input/output errors result in the output of a file information display. For files not yet opened, or for which the FOPEN intrinsic failed, this display appears as in the example below.

```
  +-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
1 !  FILE NUMBER 5       IS UNDEFINED.              !
2 !  ERROR NUMBER: 2     RESIDUE:  0       (WORDS)  !
3 !  BLOCK NUMBER: 0              NUMREC: 0         !
  +----------------------------------------------+
```

In this display, the lines indicated show the following information.

Line	Content
1	A warning that there is no corresponding file open.
2	The appropriate error number, relating to table E-4. The *residue*, which is the number of words *not* transferred in an input/output request; since no such request applies in this case, this is *zero*.
3	NOTE This will always be the last FOPEN error for the calling program. The *block number*, and *numrec* fields will always be zero in this short form.

For files that were open when a CCG (end-of-file error) or CCL (irrecoverable file error) condition code was returned, the file information display appears as shown in this example:

```
    +-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
 1  !  FILE NAME IS IN.VOLLMER.CLIFTON                !
 2  !  FOPTIONS: NEW,A,*FORMAL*,F,N,FEQ,T             !
 3  !  AOPTIONS:   INPUT,SREC,NOLOCK,DEF,BUFFER       !
 4  !  DEVICE TYPE: 0      DEVICE SUBTYPE: 9          !
 5  !  LDEV: 2       DRT: 4        UNIT: 1            !
 6  !  RECORD SIZE: 256    BLOCK SIZE: 256   (BYTES)  !
 7  !  EXTENT SIZE: 128    MAX EXTENTS: 8             !
 8  !  RECPTR: 0           RECLIMIT: 1023             !
 9  !  LOGCOUNT: 0            PHYSCOUNT: 0            !
10  !  EOF AT: 0           LABEL ADDR: %00201327630   !
11  !  FILE CODE: 0      ID IS JOE       ULABELS: 0   !
12  !  PHYSICAL STATUS: 1000000000000001              !
13  !  ERROR NUMBER: 0     RESIDUE: 0                 !
14  !  BLOCK NUMBER: 0           NUMREC: 1            !
    +-----------------------------------------------+
```

EXHIBIT N
(continued)

The lines indicated show the following information:

Line	Content
1	The *file name:* in this case, the name is IN.VOLLMER.CLIFTON
2	The *foptions* in effect, including:
	Domain: NEW = New file (as in this case). SYS = System file domain. JOB = Job temporary file domain. ALL = System *and* job temporary file domain.
	Type: A = ASCII File (as in this case). B = Binary File.
	Default File Designator: *FORMAL* = Actual file designator is same as formal file designator. $STDIN $STDLIST $STDINX $NEWPASS $OLDPASS $NULL
	Record Format: Fixed length. (as in this case) V = Variable length. U = Undefined length (as in this case). ? = Unknown format.
	Carriage Control: N = None (as in this case). C = Carriage control character expected.
	File Equation Option: FEQ = :FILE allowed (as in this case). DEQ = :FILE not allowed.
	Labeled Tape Option: T = Not a labeled tape (as in this case) L = Labeled tape
3	The *aoptions* in effect, including:
	Access Type: INPUT = Read access (as in this case). OUTPUT = Write access. OUTKEEP = Write-only access, without deleting. APPEND = Append access. IN/OUT = Input and output access. UPDATE = Update access.
	Multi-record Option: SREC = Single record access (as in this case). MREC = Multi-record access.

EXHIBIT N
(continued)

Line	Contents	
3 (Cont.)	Dynamic Locking Option:	NOLOCK = No locking permitted (as in this case). LOCK = Locking permitted.
	Exclusive Access Option:	DEF = Default specification (as in this case). EXC = Exclusive access allowed. SEA = Semi-exclusive access allowed. SHR = Sharable file.
	Buffering:	BUFFER = Automatic buffering (as in this case). NOBUFF = Inhibit buffering
4,5	The *Device Type*, *Device Subtype*, *LDEV (Logical Device Number)*, *DRT (Device Reference Table Entry Number)* and *Unit* of the device on which the file resides. (These are 0, 9, 2, 4, and 1 respectively, in this case.) If the file is a spoolfile, the LDEV will be a "virtual" rather than a physical device number. See *ldnum* under FGETINFO.	
6	The *record size* and *block size* of the offending record, in *bytes* or words as noted. (In this case, these are both specified as 256 bytes.)	
7	The *extent size* (of the current extent) and the *max extents* (maximum number of extents) allowed this file.	
8	The *recptr* (current record pointer) and *reclimit* (limit on number of records in the file).	
9	The *logcount* (present count of logical records) and *physcount* (present count of physical records) in the file.	
10	The *EOF at* (location of the current end-of-file) and the *label addr* (location of the header label of the file).	
11	The *file code*, *id* (name of creating user), and *ulabels* (number of user-created labels) for the file.	
12	The *physical status* of the file.	
13	The *error number* and *residue*, as described under the abbreviated file information display format, above.	
14	The *block number* and *numrec*, as described under the abbreviated file information display format, above.	

EXHIBIT O

OPERATIONS PROCESSING REQUEST

SYSTEM		PROCESS			
SYSTEM I	___	RUN JOB	___	REQUEST DATE	__/__/__
SYSTEM II	___	HOLD JOB	___	REQUESTED BY	_____
_____	___	OMIT JOB	___	APPROVED BY	_____
		RESTORE FILES	___	AUTHORIZED BY	_____
		OTHER	___		

JOB NAME	FILENAME	GROUP	ACCOUNT	DATE	TIME	JOB NO.
_____	_____	_____	_____	__/__/__	__:__	J#____
_____	_____	_____	_____	__/__/__	__:__	J#____
_____	_____	_____	_____	__/__/__	__:__	J#____

SPECIAL INSTRUCTIONS: _____

_____

_____

_____

DESCRIPTION OF PROCESSING (IF NOT STATED ABOVE): _____

_____

_____

_____

REASON FOR PROCESSING (IF ON SYSTEM I ONLY): _____

_____

_____

_____

OPERATORS COMMENTS: _____

_____

_____DATE __/__/__ INITIAL_____

THE LION AND THE MOUSE: HOW THE HP3000 AND PERSONAL COMPUTERS WORK TOGETHER

Robert V. Scavullo
NOESIS COMPUTING COMPANY

All of you who have read bedtime stories to your children know how the mouse was able to help the noble lion out of a deadly trap by gnawing through the net that held the lion, thereby turning the lion's scorn for the mouse into eternal gratitude. Bringing things closer to home -- for the lion read the data processing manager and for the mouse the personal computer.

This is the story of our experience at Noesis Computing Company with using personal computers in conjunction with the HP3000. We now have a great deal of respect for the personal computers. They have changed how we work and how our customers work. I have some strong feelings on where this symbiotic relationship is going -- but more on this at the end of my talk.

By way of introduction, Noesis Computing Company is an HP3000 OEM, software house and timesharing service bureau. I want to talk about both our own in-house experience and our customer's experience with personal computers. First of all, personal computers are ubiquitous. In the past 18 months, all of our HP3000 customers and all of our timesharing customers have purchased personal computers to do double duty as crt terminals and stand-alone personal computers. There are some application areas where the personal computers have completely taken over work that was done by the HP3000.

FINANCIAL MODELING

Our HP3000 timesharing service offers both DOLLARFLOW and FCS-EPS for financial modeling. Over the past 18 months all of our modeling business has migrated to personal computers. The reasons given are:

- Cost - it's less expensive to buy a personal computer than to use timesharing.

- Feel - users like the visual spread sheet style of VISICALC and MULTIPLAN.

- Capability - products like MULTIPLAN have functional capabilities that are identical to the HP3000 software packages. When used with the 16 bit large memory personal computers like the IBM PC and the HP200, the size of model you can work with can exceed the HP3000 model size.

BUSINESS GRAPHICS

Three years ago when I bought my HP four pen continuous feed plotter, I thought that I had arrived in the world of computer graphics - four different colors even! Well, I also saw that week in HP's office one of the scientific desk top graphics computers. I asked the S.E. how many colors it supported - 3,000 I believe he answered. This was my first indication that graphics on personal computers would be a hot item!

We offer two graphics software packages on our HP3000:

- SMOCK the users contributed library product that works very well for our timesharing users who do not have HP terminals, and

- DSG/3000

Our once thriving base of timesharing graphics business has deserted us for the personal computer. The HP125 with Graph/125 and the low cost two pen plotter is an unbeatable combination for both cost and capability. We still do have one customer who has his own 2623A Graphics crt and a two pen plotter. He likes the text positioning capabilities of DSG/3000.

It is worth noting that, in all the above modeling and graphics cases, the ability to easily move data from IMAGE to DOLLARFLOW and DSG/3000 was of no importance to our customers. None of their applications had any relationship with a large database.

BUT - let's talk about the successful cooperative ventures.

REMOTE OFFICE DATA ENTRY

Our star witness is one of our timesharing customers, a steamship company that uses our service to track the movement of their containers throughout the U.S. The eight regional offices report in daily on all movements of equipment in and out of their yard. Each office has at least one personal computer which is used for several tasks.

Our software on the personal computer does a very complete job of intelligent off-line data entry of equipment movements. Users work with a set of formatted crt screens with full character mode editing and extensive validation. In fact, we validate 28 out of 30 items at data entry time on the personal computer (we must wait until the data gets to the HP3000 to check if the equipment is already in the U.S. data base and whether the current transaction is consistent with the prior transaction - i.e. if the prior transaction, recorded five days ago, was a shipment from Long Beach to Chicago, the current transaction had better be a receipt in Chicago of the equipment). The local cpu lets us do some fancy footwork in changing screens and moving the cursor over certain fields based on user input values - something we would have trouble with using the HP3000 over the telephone lines.

Once or twice a day the personal computer goes on-line to our HP3000 via a leased line telephone network and burst transmits these transactions into a receiving program on the HP3000. I might add this is our only use of program to program communication - all of our other personal computer/HP3000 data transfers are merely a file transfer from one machine to the other. In the larger offices several personal computers are used for data entry while only one is linked to the HP3000. Users on the off-line work stations give their diskettes to the on-line work station for transmission.

The personal computers all serve as on-line terminals for status and activity reports which are printed daily and on-line inquiry throughout the day on the whereabouts of a particular piece of equipment.

We started this project in 1978 using Texas Instruments desk top computers with tape cassette storage and purchased conventional CPM diskette computers in 1980. From the beginning it has been a great success. The users get very high speed data entry with editing and validation usually reserved only for on-line applications - all without using any of the resources of our HP3000. The savings in telecommunications are also significant since we need only one 1200 baud line to each office instead of a more sophisticated multiplexed highspeed network.

Our network has kept up with the growth of our customer. He has added three offices in the past six months and his key offices have gone from using one to two or three work stations. All their personal computers serve only as work stations to the Equipment Control Systems. Only in the San Francisco office have they explored the idea of the personal computer as a word processor with WORDSTAR.

PROGRAMMER SUPPORT

In our office, two of our programmers have personal computers instead of regular crts. Our idea was to use the personal computer's full screen editor, (the best we saw was the UCSD editor) to create source code and then pass the code to the HP3000 for compilation and maintenance. After a several month test we dropped the idea as too cumbersome and went back to using the personal computers in dumb terminal mode with QEDIT.

I think the main reason our two programmers didn't want to bother with the UCSD personal computer editor is the beefed up capability all of our work stations have anyway. Each of our crt's are connected to a Type Ahead Engine - a sort of personal computer (it has a Z80 micro-processor and 8 kb of memory). The fully buffered I/O in the Type Ahead Engine lets the programmers type as fast as they can (this was the major argument I got for the personal computers, i.e. no typing delays) and each programmer has his own custom tool set of user defined function keys loaded with frequently typed character strings (this is old hat to you 2645 & 2624 users - but we use $800 Televideo crts). In the end the gap between a dumb crt with a Type Ahead Engine and a personal computer just wasn't that great.

CHAPTER OFFICE SUPPORT FOR ASSOCIATIONS

We do a lot of work with trade associations where the main application is membership management. Our software maintains information on members in one central on-line IMAGE database and allows the association's users easy access to the database. In the past, chapter offices usually kept their own local list of members to do their own mailings (with the predictable result that the central membership database never had the same information as the chapters). The local database was a necessity, since headquarters couldn't give them the fast response assistance they needed to meet the needs of the chapter office, to do their own mailings. To remedy this situation, we've installed personal computer work stations with correspondence quality printers and modems in each office. These work stations are used mostly as stand-alone word processors with WORDSTAR. When the chapter wants to inquire into or update a member's status they call the HP3000 and use the work station as an on-line terminal for formatted screen updates and short ad-hoc reports. When the time comes to do a mailing, working on-line they use QUIZ to select the subset of the membership database they want to mail to. This name and address file is then passed over the telephone down to a file in the chapter office personal computer where the user working off-line can create a

set of letters or notices using MAILMERGE and WORDSTAR. Local meeting planners can use a similar on-line capability to register attendees and prepare meeting schedules. Here the value of on-line updating and access to the attendee's biographical information in the central database make the on-line telecommunication worthwhile. Some of our chapter offices use their personal computers for financial planning work. None have yet developed any stand-alone local database applications.

WORD PROCESSING

Confronted by HP's wide range of word and text processing systems and software, several of our customers have opted for WORDSTAR and a personal computer instead of an HP3000 based solution. These customers focused on the need to merge names and addresses with a form letter for direct mail campaigns. They found the effort to integrate their database with a form letter was about the same using HPWORD as with WORDSTAR, MAILMERGE and a personal computer. Both require a file transfer from IMAGE into the word processing system rather than the preferred DEAR & FIRSTNAME, where FIRSTNAME is an item in the IMAGE database. Also the comparative cost of $5,000 for a HPWORD crt versus $3,500 to $5,000 for a fully equipped personal computer has been a powerful inducement to use the personal computer.

At this point I feel honor (or should I say honour as we are in Canada) bound to tell you that our secretaries still do all their correspondence and proposals with QEDIT/3000 and a homegrown formatter called RUNOFF. They looked at WORDSTAR and chose to stay with the old familiar system. We do however use WORDSTAR and MAILMERGE for our direct mail marketing campaigns.

THE FUTURE

Our biggest success with personal computers has been in giving computing support to remote locations - sort of distributed data processing on a reduced scale. I expect to see more data processing to take place independent of the headquarters' HP3000. One of our customers has approached us about building a stand-alone database application particular to the needs of one of his chapter offices - in this case - a legislative bill tracking system to be used by their lobbyist in the state capital.

At larger remote locations where more than one work station is required, I expect small multiuser systems to become very popular. ALTOS and Televideo already offer systems for 2-4 users in the $8,000 price range

that can do both the stand-alone functions and give each work station the ability to go on-line to the host HP3000. HP can't be far behind (I hope). For me as a software house, I expect the biggest breakthrough to come when a uniform operating system (front runner today is UNIX) can run on all sizes of computers from the personal computer all the way up to the large multiuser systems like a 3000/64 or a VAX. Then the data processing department will have a much easier time migrating their software to remote smaller sites. There will be no need for a data processing staff to deal with different operating systems and software suppliers for the big and small systems. Program to Program Communications has been highly touted. I have yet to set it operating in a personal computer HP3000 link for anything other than bulk data transfers. I expect the advent of the single operating system software for all size systems will make the tool easier to implement and therefore more popular.

Well, that's my report as of February 1983. I'd very much like to hear from others with experiences in this area. Perhaps, interested parties could do a panel discussion at next year's meeting.

MAKING THE MOST OF HP SOFTWARE SUPPORT SERVICES

Duane Schulz
Systems Engineer
Hewlett-Packard Company
Wilsonville, Oregon USA

INTRODUCTION

The successful development and application of user-friendly computer systems is contingent upon many factors, but the most important of these is the availability of help for the end users of these systems. In order for local data processing personnel to provide the kind of help users have grown to expect, Hewlett-Packard has developed a family of products - in fact, an entire operating division - to provide consistent help to the customer's data processing personnel: Software Support Services. Through the Systems Engineering Organization, HP provides one central service which can be crucial to the growth of our HP3000 sites to their potential - Customer Support Service, or CSS.

The CSS product, and in fact the SEO in general, is one of the most controversial and potentially misunderstood services which HP provides. The intent of this paper is to provide an informal, honest discussion of the CSS product, and how it can be used to provide you with maximum benefit in your use of the HP3000 and related systems. CSS is a contractual product, and includes clearly defined features and benefits, but at the heart of the product is a relationship between HP and its customers. Like any relationship, it works well only when those involved understand and know each other, and are willing to work together towards solutions.

I am a Hewlett-Packard Commercial (HP3000) Systems Engineer in Portland, Oregon. Prior to joining HP, I experienced HP Support firsthand as a customer (I have many years of data processing management experience). My motivation in presenting this discussion is to help as many of us as possible to take full advantage of the CSS relationship. I would like to state clearly that this is not an attempt to justify or sell CSS, and represents no statement of policy on the part of Hewlett-Packard. Because of this, the discussion will take the form of an informal presentation of ideas which come from my experiences as both an HP customer and Systems Engineer. I hope my enthusiasm for the product will not be seen as a sales effort.

To begin, we need to understand the intent of the customer support organizations at Hewlett-Packard, especially as they relate to your objectives as customers as well as HP's overall corporate objectives.

THE COMPUTER SUPPORT DIVISION

HP operates under the charter of seven clearly stated corporate objectives. From our standpoint in this discussion, the most important of these is:

> "To provide products and services of the greatest possible value to our customers, thereby gaining and holding their respect and loyalty."

Another point of orientation in understanding HP's field operation is the notion of providing Solutions (my capital) to our customers. Hopefully, you've all had the opportunity to see that, when asked to provide information on new HP products, our sales and systems engineering folks have attempted to learn more about your problem before blindly providing you with data sheets. This is because we look at things in a solution-oriented fashion - simply an extension of the objective quoted above.

Clearly, the installation of a computer system is not going to provide the solution to a business problem; the actions of the people who support the computer system is. Because of this, HP has built, over a period of years, an organization called Computer Support Division (CSD) in Cupertino, California, which has the charter of providing products and services which will assist our customers in solving their business problems by using HP computer products. Part of this division is the Systems Engineering Organization. This field operation provides a network of people who together deliver the CSS product to our customers.

Before looking at this network, it is important to note the SEO's orientation. The SEO does not develop the products you use. The development and maintenance of HP software products is the responsibility of other Divisions (HP is really a large network of small related companies). The SEO's business is rather to help make our products useful to our customers by providing help to you in the areas of sales, education, consulting, and, most importantly here, day-to-day assistance in using the software tools provided with the HP3000. Reflecting this orientation, our charter is to assist customers in making maximum productive use of goods and services provided by HP - including the need for enhancement and growth of the products.

This viewpoint has led to the development of a myriad of SEO products, but the most important tool we can use in the daily growth of our HP3000 installations is the CSS product.

This overview of HP's objectives and the intent of the SEO is important - HP is a management-by-objectives organization, and it is very important that our customers as well as members of the SEO keep these basic intentions in mind at all times when we work together. If we're all approaching our activities as steps towards the solution of a business problem, it will be rather difficult for important conflicts to develop. Indeed, when conflict does develop, it is almost always because someone has lost sight of the goals we all share.

Now that we've focused upon the basic purpose of the SEO, let's look at the physical layout of the organization which supports you through your local Systems Engineer.

THE HP FIELD OPERATION: STRUCTURE

Hewlett-Packard's field support organization is arranged as a multi-dimensional matrix of operations to provide direct access to appropriate resources. The illustration below shows the layout of the geographic entities which support you and help your Systems Engineer in delivering the full CSS product.

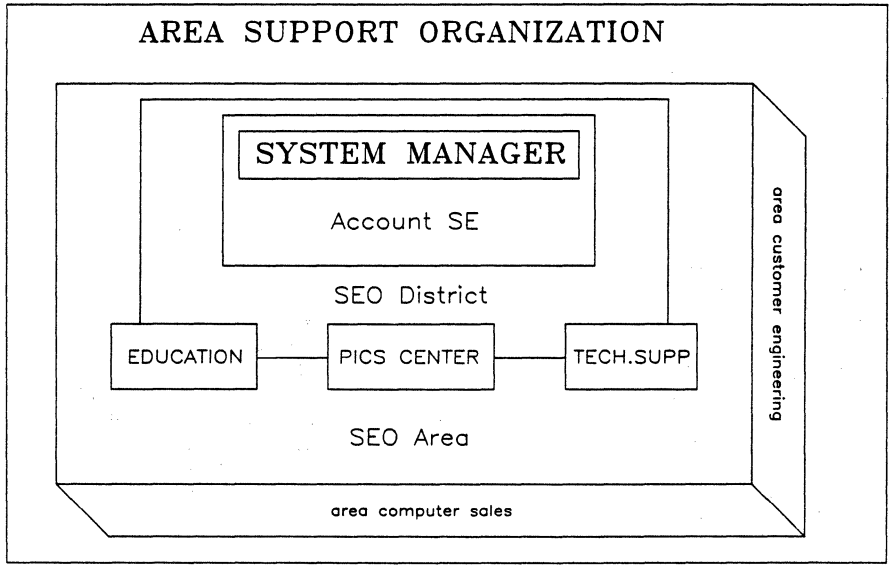


Figure 1: Area Support Layout

This illustration shows that one of the primary intents of the CSD operation is to provide assistance at the location of need. As you can see, the support network can be

viewed as a series of supportive entities surrounding every CSS customer. Let's look at each of these entities.

The System Manager is the main contact between each customer site and HP. This ensures that every site has identified an individual who will coordinate activities involving the HP3000 system. The System Manager's focus on the entire installation linked with the HP Systems Engineer's knowledge of the system's full potential offers the promise of strong growth of the system in new application areas, as well as success in all areas.

The Account Systems Engineer is given the charter of ensuring that all CSS services are delivered to each of his/her assigned customers' satisfaction. Another way of describing your SE's responsibilities is that he or she is responsible for your site's overall success in using the HP-supplied software tools you've chosen to solve your information management problems. We are held accountable for the satisfaction of each of our customers in the area of systems software; if a customer becomes dissatisfied, the SE will be responsible for remedying the situation with whatever resources are required. Finally, your SE may, depending upon your location, also perform teaching, PICS, and even technical support (specialist) functions in rotation with other SEs in the Area.

An SEO District is a group of Systems Engineers who report to a common manager in their local office. The District SE Manager is responsible for planning, organization, and leadership of District SEs in local delivery of SEO products at or above CSD standards which ensure consistent quality regardless of your location.

SEO Districts are next organized into SEO Areas. An Area is a group of Districts which is also responsible for administration of Phone-In Consulting Service (PICS), education, and technical support (product specialist) facilities throughout the Area. Together with Area Sales and Customer Engineering Managers, the Area SE Manager reports to an Area Computer Manager who is responsible for all three of these functions within the Area. Because all functions report to one individual in your Area, any problem encountered by an HP3000 customer should find resolution within the local Area. This location of resources at the source of the need is designed to provide you with a response to any sort of request for assistance quickly and with people you know or have worked with before.

Beyond this, SEO Areas are organized into Regions which finally report to the Computer Support Division, which oversees all international SEO and Customer Engineering Organization (CEO) functions.

The next illustration shows another dimension of the support matrix, this time illustrating how an SE manages resources needed to support his or her customers on a day-to-day basis.

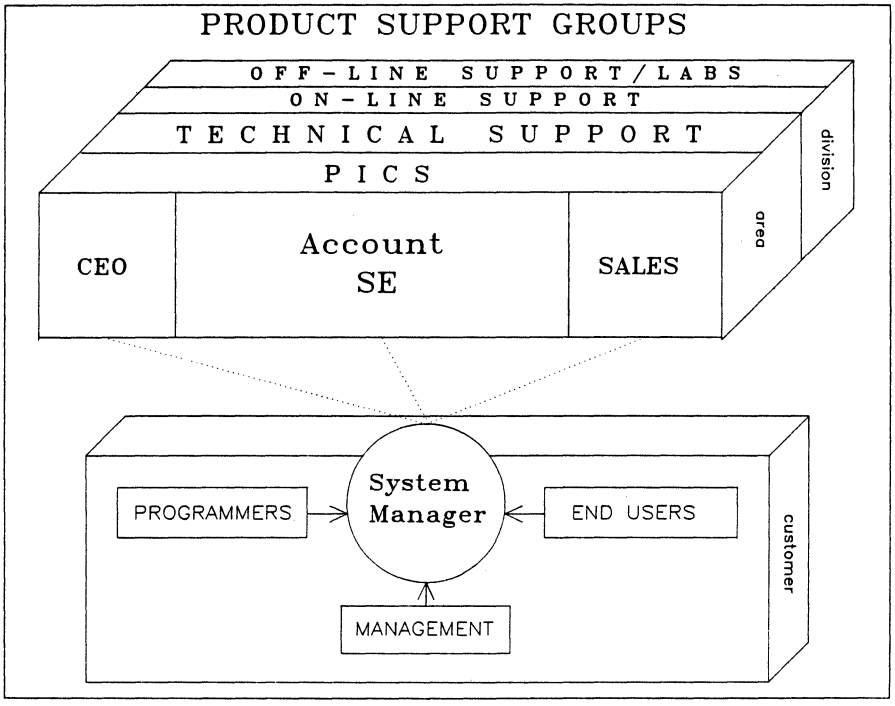


Figure 2: Levels of Product Support

Here we see that your Systems Engineer (and PICS) is actually your access to a network of support groups which lead into the software labs and product management. Naturally, this is one of the main benefits of CSS, and I can say that information and help flows smoothly both ways in the case of a good CSS relationship. We'll talk about how you might use these groups later on; first, let's look at the function of each group.

Again, your primary contact with HP is through PICS in most cases, and your Account SE in others. In the area of technical support, they are responsible for providing any information or assistance you may need to use software provided by HP, whether the need is problem resolution or information gathering.

PICS SEs are resource managers; from time to time, resolution of customer questions may require more technical product knowledge than is available in the PICS center or SEO District involved. The Area Technical Support group is an Area-local group of specialists upon whom PICS or your SE can call for help or information. In some areas, this is a dedicated team; in others (such as my Area, Northwest Neely), we have developed technical specialists throughout the SEO districts. In either case, many problems can be resolved by going to internals-trained engineers who reside in the local group; again, most problems and questions can be resolved within an Area.

When it is necessary to seek help outside the local area, or a second opinion or Division assistance is needed, an SE can look towards the On-Line Support group within the Division which produces the product. Notice that though the SEO is part of CSD, we go directly to the manufacturing division for day-to-day help on use of our products. This is an excellent example of the "matrix management" you may hear about from HP employees. In essence, On-Line Support is similar to PICS for the SEO - these folks help us through their detailed knowledge of the products. In most cases, individuals in On-Line Support will be assigned a small group of products to support, and we'll call them directly.

In some instances, On-Line Support will need to go deeper to resolve problems or provide us (and you) with the help or information we have asked for. In this instance, they have direct access to the product labs, where the code is actually written and supported. Also, Off-Line Support groups provide field training, marketing support, documentation, and many other supportive activities for each product.

Finally, Product Marketing Managers have overall responsibility for product research, development and marketing. They may need to be involved in cases such as inquiries regarding Beta-test use of new or revised products, or consideration of major enhancements.

THE FIELD SUPPORT ORGANIZATION: SUMMARY

I've included this information on the organizations which support you as a customer, through your Systems Engineer and PICS, because I have seen many instances where problems and questions could have been resolved more quickly and effectively if the "Account Team" (the System Manager and HP SE, CE, and Sales Representative) had all been aware of the depth of the organization and the groups who might be of possible help. As a customer, do you know who your Area support team members are?

I find that, as I share the form of my local organization with my customers, I provide better support, and my customers feel much less of the "we/they" syndrome which is so common with many vendor relationships. Again, when we both (HP and our customers) see that we're all working towards the same goals (solving your information problems) and understand the underlying support network, we can be much more successful at applying the products you purchase from HP.

A MODEL OF THE CSS CUSTOMER

The SEO's Customer Support Service is a product. As such, it does not fit every customer's needs. Before we proceed to the heart of this presentation, it would be worthwhile to look at a model of a "typical" CSS customer, and their use of the HP3000 system through time.

First, the customer will have a qualified System Manager working on-site (indeed, this is required by the CSS contract). Though this individual will not necessarily perform this function on a full-time basis, few HP3000 sites can be successful without easy access to a qualified system manager and data processing professional.

Secondly, the customer will probably view the HP3000 as a multifunctional information management tool which can help with various problems over the years. Though systems are frequently purchased to resolve a specific problem, such as inventory control, customers with a closed view of the computer's function within an organization will certainly not receive the benefits that a wider perspective can offer.

Finally, the customer will probably be looking at the HP3000 as a long-term investment which can evolve with the organization. An example of the potential growth of applications and uses of an HP3000 system is shown below.

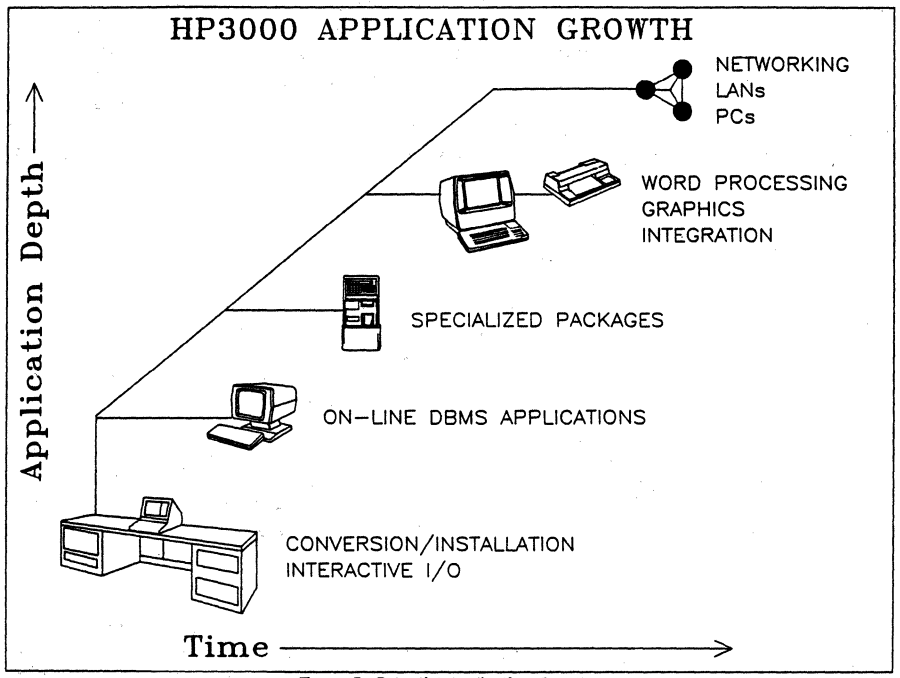


Figure 3: Potential Application Growth

In a case such as the one shown above, the customer has "grown the system" into many new areas; the HP3000 has been seen as an open-ended tool, and is being taken advantage of much more fully than perhaps was originally planned. This scenario indicates a strong need for support, not only in the area of new applications, but also in ensuring that the system continues to be accessible to all users as new applications are added. The clear need here is to provide local support which is commensurate with the increased reliance the organization is putting upon the system for overall information management.

If your application growth is even one half of that shown above, and you find that uptime and knowledge of new application areas such as office automation are becoming more and more critical, then you will probably derive the most benefit from the CSS products. Naturally, any customer who uses an HP3000 can derive benefit from CSS. However, since we at HP are taught to work to help our customers grow and solve overall information problems, the CSS product will provide even more benefit to growing customers than to static, run-only environments.

USING THE CSS PRODUCT

As I indicated above, CSS is a contractual product. The product is delivered through execution of the Customer Support Services Agreement (CSSA); CSS services are described specifically and clearly in Exhibit 2T. If you aren't familiar with the CSSA, spend some time going over it with your SE. I also find it valuable to refresh my memory by reading through the description of each service from time to time.

The remainder of this discussion will consist of a description of each of the major CSS services, including tips for the use of each service with an eye toward maximum satisfaction with the product. It is important to note that the contract clearly states what each service consists of, and outlines HP and customer responsibilities. However, being a contract, it is also worded in such a fashion as to allow for flexibility in delivery of each service. As you read through this section, please remember that each Area has the opportunity to deliver the services uniquely, and you may wish to learn more about your Area's administration of the services.

1. Account-assigned Systems Engineer

> "An HP account Systems Engineer (SE) is responsible for providing ongoing support for Customer's account, with regularly scheduled visits at Customer's site."

Your Systems Engineer is responsible for the overall success of your installation in terms of utilization of software products you have purchased from HP. Please note that this does not include software purchased elsewhere or developed using HP software products. Each SE has a unique perspective on his/her relationship with you as a customer - when your first account visit occurs with a new Systems Engineer, it is worthwhile to find out what this perspective is, and learn about how this SE interprets the services CSS provides. If you discover a discrepancy in expectations, it is far better to clarify the point before a problem arises and we are in a high-pressure situation.

In my case, I take my responsibility for my customers' success very broadly, and am willing to provide any sort of help I can, as long as it does not conflict with the CSSA, other SEO services, and my sense of propriety and ethics. For instance, if a customer needs to find other sites who use an older disc drive to discuss a problem, I can send a TWX throughout HP. In instances where a customer has not taken the opportunity to discuss these things with the account SE, CSS may end up being perceived as nothing more than an insurance policy to protect the customer from catastrophic bugs. This can and should be avoided.

The delivery of on-site account visits has evolved from Area to Area and over time. This is without question one of the two most potentially beneficial of the CSS services, and also the most misunderstood. Again, it is important to understand (ask your SE) how your Area delivers this service: how frequently are the visits delivered, how long can they last, what are they for, etc. are all important questions which frequently go unasked, with the result that we lapse into a habitual routine with these visits. Once every year, spend some account visit time deciding what you'd like to accomplish in the next year's visits. Most SEs are happy to provide various types of assistance if they are informed ahead of time.

Some of the things I've experienced are: user question-and-answer sessions; company tours to familiarize myself more fully with a total operation (and so provide more relevant help in the future); new HP product exposure (with no intent of sale - just to keep my customers aware of alternatives); brief "consultation" on data communication, DEBUG, process handling, IMAGE, application design, and about 50 other topics; and management consultation in support of the System Manager. Keep a running list of items you'd like to discuss with your SE during the next visit. The account visit does not need to be delegated only to emergencies and software installation. Our only measure is your perception of value in our help, though again we must avoid providing you with a service available through other products such as education and consulting.

2. Telephone assistance

> "Customer's System Manager will be given the telephone number for the designated HP Phone-in Consulting Service (PICS) office. By calling this number, the System Manager can contact a trained HP Systems Engineer to ask questions or seek advice relating to the use of HP-supplied software."

Like the account visit, PICS is a frequently misunderstood service. Many customers see it only as a problem- and bug-reporting service, and miss many of its major benefits. If we read the contract closely, we will see that PICS can help in areas such as application design and load management. Since your SE may frequently be unavailable for immediate contact, PICS was developed to provide you with a reliable contact point with the SEO.

Again, since PICS is administered on an Area-by-Area basis, it is worthwhile to discuss its operation with your SE, and seek advice on how to best use this service. Misunderstanding of the features of this (or any other) service can lead to frustration when a real problem arises. For instance, we guarantee a return call from a PICS SE within 4 hours, and do not guarantee resolution of every question

received at the PICS center. Learning about how this service works in your area will help you approach the service more successfully. As I've pointed out, except in cases where it is being used in place of other services such as training and consulting, it is an excellent source of assistance.

As with any such service, satisfaction with PICS support is related to communication. I've seen cases where a customer felt that a problem should be escalated to involvement of technical support specialists, but has not informed the PICS SE of this. The most important rule in using PICS effectively is to inform the person who initially logs your call of the severity of your problem. If your system is down, this should be noted; you will most likely receive a quick response from the PICS SE. On the other hand, if you're just looking for someone to chat with for a few minutes about learning the RPG/IMAGE interface, indicate this as well. Consideration and honesty are the two factors I've found most directly related to my customers' satisfaction with PICS. Last but not least, if you aren't satisfied with an SE's resolution of your PICS call, say so - we want you to be satisfied with our response.

If you understand that the PICS SE is a resource manager, you will be happy with PICS. With the number of products we support and possible rotation of PICS SEs, this is a very difficult service to provide. Be sure your account SE hears about your PICS experiences; if you have problems, your SE can resolve them or clarify your understanding of the service.

3. On-site Assistance

> "In the event that telephone assistance is not sufficient, the System Manager may request on-site assistance."

In some cases, PICS and telephone communication will not be able to resolve a problem you have encountered. If the problem seems insoluble over the telephone (and dial-up connections) and the need for on-site SE assistance has been verified by HP (this service is delivered by mutual agreement), an SE will come to your site and attempt to isolate the problem. This service is another extremely valuable part of the CSSA, and is also the most potentially misunderstood.

Every Area has the responsibility of administering some form of "escalation management", and on-site assistance is a key action in these plans. Learn how the plan works in your Area - in many cases, this service is triggered by PICS, whether the request is made by the PICS SE or your System Manager. The benefit of planned problem management is that we are able to resolve potentially emotional and

difficult to isolate problems without allowing them to get out of control.

One note of caution - this service can be abused, and is costly if it is. If your problem is the result of your application code or misunderstanding of a product, you will be liable for time and materials consulting charges. In some cases, however, this may still be more desirable than living with a difficult problem. Finally, also be aware that this service is only intended to "assist Customer in finding a workaround for Customer's software, if possible"....

Again, discuss this service with your SE ahead of time. The most important point is understanding the model before things explode. How has your Area handled recent problems requiring on-site assistance? How much advance notice has been necessary to provide on-site help? Don't forget, as a customer, you are a part of any escalation management team.

4. Software problem reporting

> "If any potential problems develop with installed HP software or updates, and the problem is not listed in the Software Status Bulletin, a Service Request may be submitted."

I am amazed at the number of people I meet who do not know about the capability to submit Service Requests to HP. This form allows customers to indicate problems, but just as importantly, it provides a vehicle for requesting product enhancements and further documentation. The SR form allows you to provide direct feedback to our labs on directions you'd like to see us follow with a product, and I can report that SRs are taken as high-priority input in the evolution of a product (in fact, as a customer, you have more weight with our product management than I do as an employee).

The key to satisfaction with the SR process is (again!) in understanding how the process is designed to work. Generally, SR forms are submitted to account SEs, who are responsible for duplication and verification of the problem (or clarification in the case of an enhancement request), and then forwarded to the On-Line Support group for the product involved. At On-Line Support, these reports are sorted by the priority indicated by the submitting SE, and prepared for the lab. Once in the lab, the problem is examined and corrected; in the case of enhancement requests, these capabilities are added to each product after bugs are fixed, according to the number of similar requests.

Be sure the problem is described in such a fashion as to allow easy duplication: isolate the problem. It will be much easier to work with a 100-line COBOL program which clearly illustrates the problem than a 2500-line program involving 2 VPLUS forms files and 18 IMAGE data sets. If

you put yourself in the position of an SE or lab programmer who is faced with a well-prepared, easy to duplicate SR and a nebulous SR with only written information, you will understand the importance of clarity and ability to duplicate. A simple-to-follow SR will elicit a quicker response in most cases.

The most notable feature of this service is its predictable nature. We are in the process of installing an on-line Service Request tracking system in the field which will allow direct inquiry on the current status of any SR. If you submit an SR, you will receive a letter from On-Line Support upon their receipt; this informs you of the status of the request. If you do not receive an acknowledgement within a reasonable period of time, call your SE - the SR may have been delayed for some reason. If you are not satisfied with the classification of or response to an SR, again call your SE; he or she should be willing to help you have the problem reclassified or reconsidered.

Once more, the handling of SR forms is a subject to discuss with your SE, since it is improved on a fairly regular basis. I find it heartening to know how much our product and lab managers rely upon customer input in the product enhancement process.

5. Software/firmware updates

> "As permanent solutions are developed for known HP software problems, they will be incorporated into planned software updates. HP will provide the System Manager with these updates as they become available."

The distribution of software updates is often a confusing issue. Generally, here are the steps for distribution of software to CSS customers: first, the MPE lab develops and tests the new release on local systems. Next, CSD performs quality testing on the software, finally releasing it to Area Field Software Coordinators (FSCs). These individuals will, at their discretion, "beta-test" the update at selected sites. When the FSCs have determined that the code is of high quality and any critical patches have been installed (this is done on a regular basis), the product is then made available to customers. One important point to note is that patch integration is done in the field, and is a benefit of CSS which is often overlooked.

Most confusion regarding software updates lies in differences in the perspective of an SE and his or her customers. In some cases, an SE may be more cautious than one or two customers, or vice-versa, and a release of MPE may be held for too long or provided too early for a specific customer. Again, this can be handled by letting your SE know where you'd like to fit into the life cycle of MPE releases:

early, average or late. Your SE should be happy to accomodate this preference.

Be sure you have an opportunity to discuss software distribution with your SE; the process changes frequently and will probably continue to do so, and there are always quite a few rumors about future releases - beware - only your Area SEO can provide you with current information.

USING CSS SERVICES: SUMMARY

Needless to say, this is a very brief discussion of many subjects which deserve papers of their own; my main intent is to provide some exposure to the nature of each service, its limitations (were you surprised at some of the contract's statements?), and suggestions for use of each service to maximum satisfaction. At best, we can use this review to develop a list of questions for our next account visits, and perhaps move forward in successful use of each service. The most frustrating experience an HP SE can have is to hear, "I never see my SE." If this is the case, it's simply time to have a talk with him or her. In many instances, a System Manager has taken over an installation which was using CSS in a fashion which the old System Manager preferred. I've experienced this myself, as both a customer and SE.

CONCLUSION

Hopefully, this broad-brush review of the CSS product has given you an opportunity to consider how you've made use of these services in the past, and perhaps provided you with some items to review with your local SEO. We all have the tendency to fall into routines in a relationship; I hope this discussion will help to shake us out of the routines we may have fallen into with CSS - this was my primary goal. To me, the beauty of the basic objectives of Hewlett-Packard's support operation, which are clearly reflected in your CSSA, is a concern for your overall success in applying our tools and an emphasis on flexibility in providing our support services.

Finally, we all need to remember that the "A" in CSSA stands for "agreement". When approached as a two-way street, CSS tends to work very well. Remember, CSS is as close as your telephone; don't forget to use it when you need to. In the end, if we remember we're all working towards the same goals and focus on flexibility and open communication, CSS can be the key to achieving these goals for all of us.

Everything You Wanted to Know about Interfacing to the HP-3000
The Inside Story

Ross Scroggs
The Type Ahead Engine Company
534 Rosal Avenue
Oakland, CA 94610
(415) 835-5603

INTRODUCTION

The subject of terminal interfacing to the HP-3000 contains no facts. None. Everything I say and you observe is an illusion supported by a lack of information and the general perversity of the universe. Maybe terminal interfacing is the fourth dimension, moving through it is certainly stranger than anything you have ever experienced before.

I have included a list of references at the end of this paper from which I have obtained some of the information included here. If you desire to make all of your terminal attachments successful, obtain all of the references and read them. The most important piece of information I can give you is to start planning early when attaching terminals to the HP-3000 and don't believe anything you read, including this paper. If you haven't seen it work yourself, plan on having to solve a few problems. This paper, derived from nine years of HP-3000 experience, is a guide to solving those problems, but it won't solve them for you.

The experiments in this paper were conducted on a Series III running the Ciper IT and a Series 44 running the Quality IT. You should expect your results will differ on different machines and IT releases.

Asynchronous terminals are attached to the HP-3000 Series I, II, and III through the Asynchronous Terminal Controller (ATC); to the Series 30, 33, 40, and 44 through the Asynchronous Data Communications Controller (ADCC); and to the Series 44 (optional) and 64 through the Advanced Terminal Processor (ATP). This paper addresses issues involved in making a successful connection to one of these three devices.

RS-232 and RS-422 are standards which describe an interface specification. They describe the electrical characteristics and control signaling conventions used by devices conforming to the standard. They do not guarantee that two RS-232 devices can communicate with each other. It is the user's responsibility to ensure compatibility of devices at the data level. The principal focus of this paper is the description on the factors that the user must control.

Terminals attached to the HP-3000 are accessed in two ways: as a session device or as a programmatically controlled device. A session device is one on which a user logs on with the HELLO or () commands and accesses the HP-3000 through MPE commands. A programmatic device is one which is controlled by an application program that is run independently from the device. These two access methods are not mutually exclusive, a session device can be accessed programmatically and many MPE commands can be executed on behalf of a user who is accessing the system programmatically.

SESSION DEVICES

Attaching a terminal as a session device is typically the easier of the two methods. You must set the terminal speed, parity, subtype, and termtype correctly and provide the proper cable to complete the hookup.

Terminal Speed

The speeds supported by the ATC are 110, 150, 300, 600, 1200, and 2400 baud. The speeds supported by the ADCC are those of the ATC plus 4800, 7200, and 9600 baud. The ATP additionally supports 19200 baud, but deletes 150 baud. Ports are either speed sensing or speed specified. Speed sensing ports automatically adjust the baud rate based upon the initial carriage return received. Speed specified ports require that the initial carriage return be received at the specified speed. Unfortunately, the ADCC can not speed sense above 2400 baud. Thus, you must log on at a lower speed and use the MPE SPEED command to access the higher speed. You can log on through the ADCC at higher speeds by utilizing speed specified ports set at the desired baud rate. The ATP will speed sense up to 9600 baud.

Terminal Parity

The format of characters processed by the HP-3000 is a single start bit, seven data bits, a parity bit, and one stop bit (two at 110 baud). The parity bit may always be zero (called space parity), always be one (called mark parity), computed for odd parity, or computed for even parity. A character with eight data bits must have no

parity bit to be compatible with the HP-3000. In this case, the eighth data bit must be set to a zero, as the HP-3000 will try to interpret is as parity even though the terminal considers it data. Choosing the proper parity setting has been complicated by differences between the ATC, ADCC, and ATP. The ATC inspects the parity bit of the initial carriage return received from the terminal and sets parity based on that bit. If the bit is a zero the ATC generates odd parity on output, if it is a one the ATC generates even parity on output. In either case the parity of incoming data is ignored and the parity bit is always set to zero before the data is passed to the data buffer. The ADCC and ATP also set parity based on the parity bit of the initial carriage return but do so with a slight, but nasty twist. If the bit is a zero, the ADCC/ATP pass through the parity bit supplied by the application program on output. If the initial parity bit is a one the ADCC/ATP generate even parity on output. If pass through parity was selected, the parity bit of the incoming data is passed through to the data buffer. If even parity was selected, the input data is checked for proper even parity and the parity bit is set to zero before the data is passed to the buffer. Thus, you should not use odd or mark parity on the ADCC/ATP. The odd parity will be interpreted as pass through and the parity bits will wind up in your data buffer, wreaking havoc. Mark parity will be interpreted as even and all input will cause parity errors.

Subtype

Subtype specifies the type of connection between the terminal and the HP-3000. The principal choices are direct connect, full duplex modem connect, and half duplex modem connect. The subtype also specifies if a terminal is to be speed sensed, or speed specified. The ATC supports subtypes 0 through 7, the ADCC supports subtypes 0 through 5; the ATP supports subtypes 0 and 1. Subtype 0 is used for directly connected terminals, no modem is used. Note that terminals that are attached to multiplexors can fit in this category, the modem involved is managed by the multiplexor, not the HP-3000. Subtype 1 is used for terminals connected through full duplex modems such as Bell 103, 212 and Vadic 34xx. Subtype 2 and 3 are used for terminals connected through half duplex modems such as the Bell 202S. Subtypes 0 through 3 speed sense on the initial carriage return. Subtypes 4 through 7 correspond to 0 through 3 with the difference being that terminals using these subtypes will not be speed sensed, they will run at a specified speed that is set at configuration time. This subtype is often used to prevent the HP-3000 from trying to speed sense garbage which sometimes occurs when using short-haul modems (line-drivers) that do not have a terminal attached to the other end. The ATC can lock out ports when this problem occurred.

Termtype

Termtype specifies the characteristics of the terminal

to be attached to the HP-3000. Most termtypes were derived from specific models of terminals that were attached to the HP in the old days, early 1970's. HP is changing this term to port protocol type. The ATC supports termtypes 0-6, 9-13, 15, 16, 18, 19, and 31. The ADCC supports termtypes 4, 6, 9, 10, 12, 13, 15, 16, 18, 19, and 31. The ATP supports termtypes 5, 6, 9, 10, 12, 13, 15, 16, 18, and 19. Termtype 4 is for Datapoint 3300 terminals, it outputs a DC3 at the end of each output line and responds to backspace with a control y, truly bizarre. (Termtype 4 on the ADCC does not output DC3s at the end of each line.) Termtype 6 is the general non-HP hardcopy terminal type. It outputs a DC3 at the end of each line but responds to a backspace with a linefeed. The linefeed is on the first backspace of a series, this allows you to type corrections under the incorrect characters. Termtype 9 is the general non-HP CRT terminal type. No DC3s are output at the end of the line (whew!!) and nothing strange happens on backspace, the cursor backs up just as you would expect. (The ATC strips out some escape sequences from the input stream that were generated by the CRT on which termtype 9 was patterned.) Termtype 10 is the general HP CRT terminal type. Termtype 13 is typically for those terminals at a great distance from the HP-3000 for which some local intelligence echos characters and the 3000 should not. (Telenet and Tymnet charge you for those echoed characters, that's reason enough not to have the HP-3000 echo them.) Termtypes 15 and 16 are for HP-263x printers. Termtype 18 is just like termtype 13 except that no DC1 is issued on a terminal read. Termtype 19 is for spooled 2361B printers. Certain termtypes less than 10 specify a delay after carriage control characters are output to the terminal. The ATC and ATP handle this by delaying for a certain of character times but do not output any characters. The ADCC actually outputs null characters. The most extreme case is termtype 6 which causes 45 nulls to be output after a cr/lf at 240 cps.

Cable

Directly connected terminals, subtypes 0 and 4, use only three signals in the cable: pin 2, Transmit Data, pin 3, Receive Data, and pin 7, Signal Ground. (Note that all signal names are given from the point of view of the terminal, not the modem or the HP-3000 which acts like a modem.) Typically the cable will connect pin 2 at the terminal end to pin 2 at the HP-3000, pin 3 at the terminal to pin 3 at the HP-3000 and pin 7 at the terminal to pin 7 at the HP-3000. This is not to say that your terminal does not require other signals, it just says that the HP-3000 is not going to provide them, you must. If your terminal requires signals like Data Set Ready, Data Carrier Detect, or Clear To Send, you can usually supply these signals to the terminal with a simple cable patch. Jumper pin 4, Request To Send to pin 5, Clear To Send. Jumper pin 20, Data Terminal Ready to pin 6, Data Set Ready and pin 8, Data Carrier Detect. These two jumpers cause the terminal to supply its required signals to itself.

Modem connected terminals, subtypes 1 and 5, use seven signals in the cable: pin 2, Transmit Data; pin 3, Receive Data; pin 4, Request To Send; pin 6, Data Set Ready; pin 7, Signal Ground; pin 8, Data Carrier Detect; and pin 20, Data Terminal Ready. Naming the signals gets complicated since the HP-3000 is acting like a modem and it is being attached to a modem. Typically, the cable that connects the HP-3000 to the modem will connect pin 2 at the modem end to pin 3 at the HP-3000, pin 3 at the modem to pin 2 at the HP-3000, pin 4 at the modem to pin 8 at the HP-3000, pin 6 at the modem to pin 20 at the HP-3000, pin 7 at the modem to pin 7 at the HP-3000, pin 8 at at the modem to pin 4 at the HP-3000, and pin 20 at the modem to pin 6 at the HP-3000.

You should note an important characteristic of the cable descriptions given above. The terminal to HP cable was "straight-through,", like-numbered pins were connected together. The modem to HP cable was a "cross-over," pairs of pins were connected crossed. Why the difference? The explanation is that the world is divided into Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). A DTE is terminal or something like it which sits at the end of a data communication line. A DCE is a modem or something like it which is part of the data communication line. The distinction is not rigid, the HP-3000 acts like a DCE. Multiplexors may look like a DCE to the terminals attached to it, and it may look like a DTE to the modem to which it is attached. The cabling principle is that a DTE to DCE connection uses a straight through cable and a DTE to DTE or DCE to DCE connection uses a cross over cable.

The cable that attaches your terminal to a modem should be specified in your terminal owners manual, consult it for proper connections.

Flow Control

Flow control is the mechanism by which the rate of data flow between the HP-3000 and the terminal is controlled. The HP-3000 supports two output flow control methods, ENQ/ACK and XON/XOFF. The HP supports one input flow control protocol, DC1/DC2/DC1, commonly referred to as the "block mode" protocol.

The ENQ/ACK protocol is controlled by the HP-3000. After every 80 characters output the system sends an ENQ to the terminal and suspends further output until and ACK is received back from the terminal. The suspension is of limited duration for termtypes 10 to 12, output resumes if no ACK is received in a short amount of time. The suspension is indefinite for termtypes 15 and 16, the ENQ is repeated every few seconds until an ACK is received.

It is the ENQ/ACK protocol that fouls up non-HP terminals that attempt to access the HP-3000 through a port that is configured for an HP terminal. Most terminals do

not respond to an ENQ with an ACK, you must do it manually by typing control-f which is an ACK. An ENQ is generated by the HP-3000 when the initial carriage return is received from the terminal, thus you get hung immediately. But, hit control-f, and logon and specify the proper termtype in your HELLO command. The ATP does not output the initial ENQ upon receipt of the carriage return, so your non-HP terminal will not receive this initial ENQ, but you must still specify the proper termtype.

The XON/XOFF flow control protocol is controlled by the terminal. When the terminal wishes to suspend output from the HP-3000 it sends an XOFF (control-s or DC3) to the HP-3000 and sends an XON (control-q or DC1) to resume output. Unfortunately the HP-3000 sometimes fails to properly handle one of the two characters and you either overflow your terminal or get hung up. This is particularly nasty when your terminal is a receive-only printer and you can't supply a missing XON. You're really dead if the HP-3000 misses the XOFF. XON/XOFF is not handled well by the ADCC and ATP. Neither controller strips the parity bit of incoming characters when pass-thru parity is in effect. Thus, a terminal using any parity other than space will have all of its XONs or XOFFs ignored since the character with the parity bit included will not match the character used by the controller which has its parity bit set to zero. The ADCC ignores all XOFFs and XONs when no output is active. Thus, if the last character output is the one that causes your terminal to generate an XOFF, the ADCC tosses the XOFF and proceeds to overrun your terminal.

A special note on XON. If you inadvertently send an XON (DC1) to the HP-3000 when output is not suspended, you are now in paper tape mode and backspace, control-x, and linefeed will act strangely. Hit a single control-y to get out of this mode.

Some terminals perform flow control by raising and lowering a signal on their interface, the HP-3000 can not handle this. You must either run the terminal at a low enough speed to avoid overflowing it or provide hardware to convert the high/low signal to ENQ/ACK or XON/XOFF, a costly affair.

The form of flow control used by HP terminals when block mode is enabled is the DC1/DC2/DC1 protocol. When the enter key is pressed on the terminal, a DC2 is sent to the HP-3000 after receipt of a DC1 to alert the 3000 of a pending block mode transfer. When the HP-3000 is ready to receive the data it sends a DC1 back to the terminal to start the data transfer. (Your program does not handle the DC2/DC1, but see below FCONTROL 28, 29.) This works fine except in certain circumstances. In certain modes the terminal actually sends DC2 carriage return when the enter key is pressed. This is no problem unless the DC2 and CR do not arrive at the HP-3000 together. The CR may be seen as the end of the data if it comes sufficiently far behind the DC2, your program completes its request for data with nothing and

the real data bites the dust when it finally shows up. The separation of the DC2 and CR can occur when using statistical multiplexors or when using Telenet or Tymnet. Be aware, this problem is infrequent, but unsettling when it occurs. The ATP attempts to solve this problem by deleting any carriage return that follows a DC2, regardless of the distance between them.

Special Considerations

Every shop should have the proper tools to perform its tasks. Don't neglect your terminal needs. Keep on hand a small flat blade screwdriver, a small Phillips head screwdriver, needlenose pliers, and some sort of breakout box. A breakout box is a small box which is placed inline between two devices. It allows you to monitor, via leds, certain RS-232 leads. This lets you visually verify the state of modem signals and whether data is actually flowing. It allows recabling by providing a jumper area so that any pin to pin combination desired is achievable. These boxes cost from $38 to $200, even HP re-markets one. Consult the John Beckett, John Kendall, reference given below for poor man's data comm tools.

The RS-232 specification requires that terminals running at high speed be no further than 50 feet apart. Almost everyone ignores this specification and experiences no problems. ATP users are beginning to have the rule enforced, the ATP seems more sensitive to long RS-232 lines. Asynchronous line drivers (short haul modems) can be used to drive terminals at greater distances. The HP implementation of RS-422 specifies a maximum separation of 4000 feet at high speeds. This improvement, as well as the fact that the interface is almost immune to noise, will simplify certain aspects of terminal interconnection in the future. Unfortunately, the two specifications are incompatible and switching from RS-232 to RS-422 is not trivial.

A multiplexor is a device that allows many terminals at a remote location to be connected to the HP-3000 over a single communication line. Each terminal is connected to a distinct port on the HP-3000, but savings are realized because all terminals share the phone line. Multiplexors attempt to maximize the shared use of the line, but in doing so have to use flow control protocols if the aggregate data rate from the terminals or computer exceeds that of the data communicaton line. Multiplexors can be setup to use XON/XOFF flow control protocol at either end. On the computer end this usually causes little trouble. On the terminal end, though, problems abound. Neither the user, typing in character mode, or the terminal, transmitting in block mode, is likely to obey an XOFF. Many block mode terminals sharing a line may overflow it if sufficient capacity is not available. The ENQ/ACK protocol does not perform well with some multiplexors. The delay between the transmission of the ENQ from the HP-3000 and the subsequent receipt of the ACK generated by the terminal can cause the

terminal to print in a start stop mode. Some multiplexors attempt to solve this problem by emulating the ENQ/ACK protocol at each end. The multiplexor at the 3000 end responds to ENQ with an ACK and the multiplexor at the terminal end generates an ENQ every 80 characters and waits for the ACK response. This feature usually allows the transmission to proceed more more smoothly.

The value added networks, Tymnet and Telenet, also use XON/XOFF and the same problem with sending an XOFF to the terminal exists as it does with multiplexors. The networks can be configured not to use XON/XOFF, buy you can still lose data.

A port selector is a device that allows many terminals to be connected to not-so-many computer ports. Terminals are assigned to ports on a first-come, first-served basis until all ports are consumed. Subsequent terminal service requests are refused or held until a port becomes available. Beyond this basic operation, port selectors can implement priority schemes, allocate terminals to ports on different computers, and perform automatic disconnects. One question arises though. How does the port selector know when a port on the HP-3000 is available? When a terminal logs off a direct connect port, subtype 0 or 4, there is no indication other than the logoff message that the port is free. Use of subtype 1 or 5 for ports controlled by port selectors causes a modem signal to drop on logoff which can be used by the port selector as an indiction that the port is free. Some systems require that all terminals provide a terminal ready modem signal. When the terminal is powered off, the signal is lowered and the port to which the terminal was connected is considered free.

PROGRAMMATIC DEVICES

Attaching a terminal as a programmatic device is usually done when you want to attach a serial printer, instrument, data collection device, or other strange beast to the HP-3000. An application program you write will typically control all access to the device; a user will not walk up to it, hit return, and log on. I will explain the various intrinsics that are used to access programmatic devices.

FOPEN

You must call FOPEN to gain access to the device. I always use a formal file name to allow control of the open with file equations. If the device is unique in the system, I use its device name as the file name. The foptions specify CCTL, undefined length records, ASCII, and a new file. The aoptions specify nobuf, exclusive access and input/output. Choose a record size that is larger than the maximum data transfer that will take place.

For devices that are to be used exclusively in programmatic mode it is recommended that you REFUSE the device so that extraneous carriage returns from the device will not be interpreted as logon attempts by the HP-3000.

Opening subtype 1 or 5 ports differs among the controllers regarding the point at which you hang if the controller finds that the modem is not online. Programatically handling incoming calls can be tricky, experiment carefully.

FCLOSE

You call FCLOSE to release access to the device. Though FCLOSE resets most FCONTROL options, it is good practice to explicitly reset all FCONTROL options before calling FCLOSE.

ATC - MPE sends a cr/lf to the device if it believes that the "carriage" is not at the beginning of the line, i.e., the last character output was not a linefeed.

ADCC - MPE sends a cr/lf to the device if it believes that the "carriage" is not at the beginning of the line, i.e., the last character output was not a linefeed or formfeed.

FREAD

You call FREAD to get data from the device. Many of the FCONTROL calls shown below affect how FREAD works. End-of-file is indicated by a record that contains ":EOF:". Any record with a colon in column one is an end-of-file to $STDIN. ":EOD", ":EOJ", ":JOB", ":DATA", and ":EOF:" are end-of-file to $STDINX.

The default end of record terminator is carriage return. You should change this if the device terminates all records with some other character. You can specify an alternate terminator that will terminate a record in addition to carriage return. Choose the teminator so that it is the last character input. For a device that sends a linefeed after carriage return, try to use linefeed as the terminator instead of carriage return. See FCONTROL 41 below.

Some devices send data followed by a fixed terminator followed by a LRC or CRC character. This error checking character can take on all values, thus it can not be used as a terminator. Unfortunately, it often looks like XOFF which will halt any further output to the device on ATC or ATP controllers. (A bug, I think so, in the ADCC saves you since it doesn't recognize XOFF except during output.)

You may want to trap certain errors returned by FREAD to your program: 22, software time-out; 31, end of line (alternate terminator); and 28, timing error or data overrun. This last error occurs frequently on ADCCs running at high speeds.

ATC - The characters NULL, BS, LF, CR, DC1, DC3, CAN (control-x), EM (control-y), ESC:, ESC;, and DEL are stripped from the input stream for both session and programmatic devices.

ADCC, ATP - The characters BS, LF, CR, CAN (control-x), and EM (control-y) are stripped from the input stream for session devices. The characters BS, CR, and CAN (control-x) are stripped from the input stream for programmatic devices.

Each time you issue an FREAD to the terminal MPE sends a DC1 to the terminal to indicate that it is ready to accept data. Most devices ignore, totally, the DC1. If your a device reacts negatively to the DC1, use termtype 18 which suppresses the DC1 on terminal reads. The device must not send data to the HP-3000 until it has received the DC1, otherwise the data will be lost. If the device does not wait for the DC1 you must supply external hardware that will provide buffering and wait for the DC1 or you can solve the problem on the HP-3000 by using two ports to access the device. One port is opened for reading and the other for writing. A no-wait read is issued before the write that causes the device to send data, then the read is completed. When you attach your device to the two ports, connect pin 2, Transmit Data of the terminal to pin 2 of the read port, connect pin 3, Receive Data of the terminal to pin 3 of the write port, and pin 7, Signal Ground of the terminal to pin 7 of both ports. (This two port scheme was first introduced to me by Jack Armstrong and Martin Gorfinkel of LARC.)

FWRITE

You call FWRITE to send data to the device. The carriage control (cctl) value of %320 is often used to designate that MPE send no carriage control bytes, such as cr/lf, to the device. Some FCONTROL calls shown below affect how FWRITE works. Control returns to your program from FWRITE as soon as the data is loaded into the terminal buffers, it does not wait until all data has been output to the device.

ATC - Carriage control %61 is output as carriage return, formfeed (termtype 10).

ADCC - Carriage control %61 is output as formfeed (termtype 10).

ATC, ADCC - the initial FWRITE to an HP terminal termtype causes an ENQ to be sent to the terminal.

ATP - the initial FWRITE does not send an ENQ.

FSETMODE - 4 - Suppress carriage return/linefeed

In normal operation a line feed is sent to the terminal

if the input line terminates with a carriage return, a cr/lf is sent to the terminal if the line terminates by count, and nothing is sent if the line terminates with an alternate terminator. These extra characters may not de desirable in certain applications. FSETMODE 4 suppresses these linefeeds and carriage returns. FSETMODE 0 returns to normal line termination handling, an FCLOSE also returns the device to the normal mode.

FCONTROL

FCONTROL is the workhorse intrinsic for managing a programmatic device on the HP-3000. Each use of FCONTROL which be shown separately but it will usually be the case that several calls will be used. Most calls are required only once, but the timer calls are required for each input operation. Each call will be identified by the controlcode parameter that is passed to FCONTROL.

FCONTROL - 4 - Set input time-out

This option sets a time limit on the next read from the terminal. It should always be used with devices that operate without an attached user to prevent a "hang". If something goes wrong with the device, your program will not wait forever, control will be returned eventually. The FREAD will fail and a call to FCHECK will return the errorcode 22, software time-out. No data is returned to your buffer in the case of a time-out, any data entered before the time-out is lost. If you issue a timeout for a block mode read the timer is stopped when the DC2 is received from the terminal, a new timer is then started which is independent of the timer set by this FCONTROL call. See the section below on enabling/disabling user block mode transfers. The ATP does not terminate your timer on receipt of the DC2. If this special timer terminates the read, FCHECK will return error 27.

FCONTROL - 10, 11 - Set terminal input/output speed

These FCONTROL options allow you to change the terminal input and output speeds. FCONTROL 37 can also be used to set terminal speed, it sets termtype as well and is the method that I prefer.

ATC - Split speeds are allowed.

ADCC, ATP - Split speeds are not allowed, FCONTROL 10 and 11 set both input and output speed.

FCONTROL - 12, 13 - Enable/disable input echo

These FCONTROL options allow you to enable and disable

terminal input echoing. Many devices that attach to the HP-3000 do not expect or desire echoing of the characters they transmit. This option along with FSETMODE 4 completely turns off input echoing. Echoing is not restored when a file is closed so you should always put echo back the way it was found.

FCONTROL - 14, 15 - Disable/enable system break

The break key should be disabled if terrible things will happen when the user hits break and aborts out of a program. You, the programmer, always seem to need break for debugging purposes and discover that you have it turned off. System break can only be enabled for session devices, it is not allowed for programmatic devices. If break is entered on a session device the data already input will be retained and provided to the user program after a resume and the completion of the read. If a break is entered on a programmatic device a null will be echoed to the device but no data is lost.

FCONTROL - 16, 17 - Disable/enable subsystem break

Subsystem break is recognized only on session devices, it can be enabled on programmatic devices but has no effect. If a control-y is entered during a read, the read terminates and the data already input will be retained and provided to the user program after the control-y trap procedure returns. If control-y is disabled, any control-y will be stripped from the input but no trap procedure is called and the read continues. Control-y trap procedures are armed by the XCONTRAP intrinsic. A subsystem break character other than control-y may be specified when unedited terminal mode (FCONTROL 41) is used.

ATC - In programmatic mode control-y's are always stripped from the input.

ADCC - In programmatic mode control-y is not stripped from the input if subsystem break is enabled.

FCONTROL - 18, 19 - Disable/enable tape mode

ATC, ADCC - This is effectively an FSETMODE 4, an FCONTROL 35, and suppression of backspace echoing all rolled into one.

ATP - Tape mode can not be enabled.

FCONTROL - 20, 21, 22 - Disable/enable terminal input timer, read timer

These options can be used to determine the length of time it took to satisfy a terminal read. It is not a

time-out, that is FCONTROL 4. The manual states that you must enable the timer before each read so why is there a disable option? If you read the timer without enabling the timer, you get the time of the most recent read that did have the timer enabled. The number returned is the length of the read in one-hundreths of a second.

FCONTROL - 23, 24 - Disable/enable parity checking

This option enables parity checking on input for the parity sense specified by FCONTROL 36.

ATC - This option affects input parity checking only, output parity generation is controlled by FCONTROL 36.

ADCC, ATP - This options controls both input parity checking and output parity generation, FCONTROL 36 only specifies the type of parity.

FCONTROL - 25 - Define alternate line terminator

This option is used to select an alternate character that will terminate terminal input in addition to carriage return. It is useful if your device terminates input with something other than return.

ATC - Backspace, linefeed, carriage return, DC1, DC3, control-x, control-y, NULL, and DEL are not allowed as terminators. The manual claims that DC2 and ESC are not allowed as terminators but they work. If a DC2 is the first input character from an HP termtype terminal the HP-3000 drops the DC2 and sends a DC1 back to the terminal, it thinks a block mode transfer is starting. Any other DC2 is recognized as a terminator if enabled. By enabling user block mode transfers (FCONTROL 29), a DC2 as the first character will also be recognized as a terminator when enabled. For non-HP termtype terminals a DC2 is always recognized as a terminator when enabled.

ADCC - Backspace, linefeed, carriage return, control-x, control-y, and NULL are not allowed as terminators. The manual claims that DC1, DC3, ESC, and DEL are not allowed as terminators, but they work. DC2 is allowed as a terminator but produces bizarre results unless unedited terminal mode (FCONTROL 41) is also enabled in which case the DC2 is recognized as a terminator in any position.

ATP - everything is allowed as an alternate terminator, even carriage return. (Carriage return as an alternate to itself does not perform quite right. It acts somewhat as a normal terminator, somewhat as an alternate. But why do it?)

If a line terminates with the alternate terminator, the character will be included in the input buffer and counted in the length. A read terminated by the alternate character always returns an error condition. You must call FCHECK to

determine that the read terminated with the alternate character.

FCONTROL - 26, 27 - Disable/enable binary transfers

Binary transfers can be used to transmit full 8-bit characters to and from the terminal. On input, a read will only be satisfied by receiving all characters requested, a carriage return or alternate terminator will not terminate the read. Thus, you must always know how many characters to read on each input from the terminal. Enabling binary transfers also turns off the ENQ/ACK flow control protocol and carriage control on output. No special characters are recognized on input. See the note under FCONTROL 25 about DC2 as the first input character on a line. No cr/lf is echoed to the terminal at the end of the read. If a session device is being accessed in binary mode, a break will remove the terminal from binary mode but it will not be returned to binary mode when a resume is executed.

ATC, ATP - binary transfers overrides parity checking.

ADCC - parity checking overrides binary transfers.

FCONTROL - 28, 29 - Disable/enable user block mode transfers

As described above, the normal sequence of events in a block mode transfer from an HP terminal to the 3000 is for the HP-3000 to send a DC1 to the terminal indicating its readiness to accept data, the terminal sends a DC2 when the enter key is struck to indicate that it is ready to send data, the HP-3000 responds with another DC1 when it is really ready to take the data, and the terminal sends the data. All of this is transparent to your program which just issues a big read. If your would like to participate in this handshake you enable user block mode transfers and MPE relinquishes control of the handshake. Your program would issue a small read, get the DC2, and issue another read to accept the data. This allows you to meddle around before the data shows up.

The terminal driver only supports block mode transfers with HP termtypes and performs one other function during block mode transfers. Normally you wouldn't put a timeout (FCONTROL 4) on a block mode read because the user can take an indefinite amount of time to fill a screen; but you would like to avoid terminal hangs because data or the terminator from the terminal gets lost. This situation is handled by the driver for you, the portion of the read after the second DC1 is sent to the terminal is timed for (#chars in read/#chars per sec)+30 seconds. If data or the terminator is lost and the read times out, the read will fail and FCHECK will return error 27. The ATP does not perform this function unless FCONTROL 31 is in effect.

This FCONTROL is useful to prevent the ATC and ATP from

recognizing a leading DC2 during binary or unedited transfers. These controllers have a strong desire for block mode and will interpret leading DC2s, and embedded DC2 CR, as block mode triggers. FCONTROL 27 prevents the controller from ever recognizing DC2 as a special character.

FCONTROL - 30, 31 - Disable/enable V/3000 driver control

This option is an undocumented option in which the terminal driver provides low level support for V/3000 use of terminals. When V/3000 issues a read to the terminal the driver outputs a DC1; the terminal user hits enter which causes a DC2 to be sent to the 3000; the driver responds with ESC c ESC H DC1 which locks the keyboard and homes the cursor; it appears that the driver also enables binary transfers because the second read only terminates by count, not by terminator. The portion of the read following the second DC1 is timed as described under FCONTROL 28, 29. (The ATP sends ESC H ESC c, just a little inconsistency to keep you awake.)

FCONTROL - 34, 35 - Disable/enable line deletion echo suppression

This option suppresses the !!!cr/lf echo whenever a control-x is received from the terminal, the control-x still deletes all data in the input buffer.

FCONTROL - 36 - Set parity

This FCONTROL option sets the sense of the parity generated on output and checked on input. The four possibilities are: 0, space or no parity, all 8 bits of the data are passed thru; 1, no parity, the parity bit is always set to one; 2, even, even parity is generated on all characters; and 3, odd parity, odd parity is generated on all characters.

An undocumented effect of this FCONTROL call is that the previous parity setting is returned in the param parameter wiping out its original value!

ATC - FCONTROL 36 sets the parity sense and enables output parity generation. FCONTROL 24 must be called to enable parity checking on input.

ADCC, ATP - FCONTROL 36 sets the parity sense only. FCONTROL 24 must be called to enable output parity generation which results in input parity checking as well.

On the ATC, parity is not reset to the default when a device is closed. This can be useful if you have a session device that can not run with the default parity. Each time the system is started run a program that opens the device, sets the parity, and closes the device. It can then be accessed

as a session device with the required parity.

The following tables shows the results of testing the various parity options. Check these yourself, some experiments produced such bizarre results that they can not be reported here. In each case, both FCONTROL 24 and FCONTROL 36 were specified so that parity generation was enabled on output and parity checking was enabled on input.

Option 0 - Space parity or parity pass through

ATC - generated space parity on output, did no checking and stripped parity bits on input.

ADCC, ATP - generated space parity on output, checked for even parity on input.

Option 1 - Mark parity

ATC - generated mark parity on output, did no checking and stripped parity bits on input.

ADCC - generated odd parity on output, checked for odd parity on input.

ATP - generated mark parity on output, checked for odd parity on input.

Option 2 - Even parity

ATC, ADCC, ATP - generated even parity on output, checked for even parity on input and stripped parity bits.

Option 3 - Odd parity

ATC, ADCC, ATP - generated odd parity on output, checked for off parity on input and stripped parity bits.

FCONTROL - 37 - Allocate a terminal

In the old days you had to allocate a programmatic terminal before it could be used. Now you don't even though the manual claims that you do. This option is still useful because it allows you to set the termtype and terminal speed with one FCONTROL call. Common sense, mine at least, says to set termtype and speed each time a device is opened even if the proper values are configured in the i/o tables. Using this option allows use of a file equation redirecting the program to another device that might not be properly configured.

FCONTROL - 38 - Set terminal type

This option allows you to set the terminal type, but use FCONTROL 37 and set type and speed all in one shot.

FCONTROL - 39 - Obtain terminal type information

Before changing the terminal type, get the current value and reset it when you are through.

FCONTROL - 40 - Obtain terminal output speed

Before changing the terminal speed, get the current value and reset it when you are through.

FCONTROL - 41 - Set unedited terminal mode

Unedited terminal mode is about the most useful FCONTROL option used to communicate with programmatic devices. It allows almost all control characters to pass through to the HP-3000 but does not require reads of exact length as in binary transfers. Input will terminate on a carriage return or an alternate terminator if specified. The subsystem break character, replacing control-y, can also be specified.

ATC - Unedited terminal mode overrides input parity checking, no checking is performed and all input parity bits are set to zero. Output parity generation is performed normally.

ADCC, ATP - parity checking takes precedence over unedited mode.

Binary transfers enabled overrides unedited terminal mode enabled. If the input terminates with the end-of-record character or alternate terminator no cr/lf is sent to the terminal. If the input terminates by count a cr/lf is sent to the terminal unless an FSETMODE 4 has been done. Unedited mode does not turn off the ENQ/ACK flow control protocol on the ATC or ADCC. See the note under FCONTROL 25 about using DC2 as a terminator.

PTAPE

The manual describes PTAPE as the intrinsic to use to read paper tapes. (A fancy data-entry media that is becoming increasingly popular.) It can be used on the HP-3000 to access devices that send up to 32767 characters all in one shot subject to a few limitations. The data must be record oriented with carriage returns between records, MPE will cut the data into 256 character records if there are no returns, and the whole mess must be terminated by a control-y. Certain buffering terminals allow you to fill their memory off-line, connect to a computer, and transmit all the data. This could save considerable time and money over dial-up phone lines.

DEBUGGING

If you have a requirement to attach a programmatic device to the HP-3000, the worst strategy is to write some code on the 3000, plug the device in and start testing. Murphy says it won't work and it won't. The method I use is to test the device, then the code, and then the code and device together. I test the device by plugging it into an HP-2645 (or equivalent) terminal, turning on monitor mode, and simulate the HP-3000 by typing on the keyboard. (Remember that you are hooking two terminals together, you will probably hook device pin 2 to 2645 pin 3, device pin 3 to 2645 pin 2, and device pin 7 to 2645 pin 7.) You can stimulate the device and observe all responses quite simply. Any strange behavior can be noted at this point. The next step is to write the code on the HP-3000 to access the device in the manner determined by the first tests. Then plug the HP-2645, not the device, into the HP-3000. Now type on the 2645 to simulate the device, continue until your code is debugged. Now you can plug the device into the HP-3000 and you have a good (modulo Murphy) chance of actually getting it to work.

REFERENCES

Communications Handbook, Hewlett-Packard Company, April 1981 Part # 30000-90105 This manual supersedes the HP Guidebook to Data Communications and the Data Communications Pocket Guide

Don Van Pernis, "HP-3000 Series II Asynchronous Terminal Controller Specifications", Computer Systems Communicator, #15, December 1977, page 2. This explains the terminal subtypes and signal requirements for the ATC.

Charles J. Villa, Jr., "Asynchronous Communications Protocols", Journal of the HP General Systems Users Group, Volume 1, #6, March/April 1978, page2. Good introductory material.

John Beckett and John Kendall, "Poor Man's Data Communications", Interact, Volume 2, #6, November/December 1982, page 14. Good, inexpensive data comm tools.

Tom Harbron, "Lightning, Transients and the RS-232 Interface", Journal of the HP General Systems Users Group, Volume III, #3, Third Quarter 1980, page 14. How to avoid being zapped.

Black Box Catalog, P.O. Box 12800, Pittsburgh, PA 15241, (412) 746-2910 A catalog that is required in every shop.

MPE Intrinsics Manual, Hewlett-Packard Company, January 1981 Part # 30000-90010 Chapter 5 discusses most of the FCONTROL options that are applicable in terms of the ATC, it is often inaccurate in describing the ADCC.

Making The User Feel Pampered: How To Do It and Why It Pays Off

Christopher C. Sieger
The Baltimore Spice Company

Are you getting a little tired of the term "user-friendly"? Doesn't it seem to have joined the pantheon of overworked buzzwords of which we in data processing are so proud? It is by now a candidate for the DP Jargon Hall of Fame, along with such all-time favorites as "Structured System Design," "Modular Programming," etc. Indeed, the term isn't any longer the fastest rising cliche on the horizon; the closely related word "ergonomics" must be considered for that honor.

However, words or phrases generally don't become cliches unless they have been proven useful and/or true over a significant period of time. Just as the current emphasis on structured programming/analysis/requirements definition/design has its roots in the recognition of the need to adhere to a coherent set of principles in the generation of systems and programs, as well as some tacit agreement among professionals regarding the key elements in the set, the concepts of "user-friendly," "ergonomically-designed" systems stem from general recognition of the fact that users are part of the system being analyzed, not something outside the scope of analysis. In other words, these terms are based on common sense.

Nevertheless, the fact that these terms have attained the status of generally accepted principles of system design does not mean that everybody pays serious attention to them. After all, the fact that almost every programmer in the business today claims to know what structured programming is all about does not mean that we have stopped producing poorly designed programs. Similarly, although almost everybody in our business purports to recognize the importance of the user's role in both systems development and operation, the design of data processing systems is often relegated to MIS "experts," as far removed from the messy confines of reality as possible. Once implemented, the operation of these systems falls to these same DP people. In many cases the data processing department has come to the conclusion that their own people know more about how the company does business - or at least about how the company should do business - than the users whose everyday task is to fulfill the main purpose of the company's existence. Even in cases where the situation has not deteriorated to that extent, there is still a feeling among some DP people that the department has a life and purpose of its own. In fact, no DP department has any excuse for existence other than the support of those in their company who are engaged in the

primary business of their organization, be it manufacturing widgets or writing government contract reports. The DP department is a support organization.

One result of DP's arrogant attitude has been the creation of an adversary relationship between many DP departments and those whom they were intended to serve. In many organizations, the data processing department is both disliked and distrusted by "line" departments. The technical, esoteric, even mysterious nature of the workings of computers and their keepers is regarded by many of the uninitiated as extremely threatening.

And we foster this attitude. Whether it stems from a desire to be recognized as a member of an elite class or the fear that somebody might discover that we possess no magic formulas, we go to extremes to let users know that we are not part of the same unwashed mass to which they belong. Some people, however, can see our feet of clay:

> First get it through your head that computers are big, expensive, fast, dumb adding-machine-typewriters. Then realize that most of the computer technicians that you're likely to meet or hire are complicators not simplifiers. They're trying to make it look tough. Not easy. They're building a mystique, a priesthood, their own mumbojumbo ritual to keep you from knowing what they - and you - are doing.[1]

For the most part, the gulf of antagonism and suspicion between us and our users is our fault - those of us who regard ourselves as data processing professionals. We forget that people are the system. When we design or redesign business systems or manufacturing systems, our prejudices lead us to center our design around the computer, not people. We have come to regard the speed of the computer as a panacea for all our problems; unfortunately this often leads us to use the computer as a crutch, a means of treating a symptom rather than a problem. Until we refocus our efforts on people rather than machines, alienation from our users is inevitable. Further, unless our users feel that we are part of their team, working with them, assisting them toward their primary goals, we will not have their support when we need it, and our systems will suffer accordingly.

We as data processing professionals must start with the proper attitude toward our users. This attitude should come very close to the old adage, "The customer is always right." Our users must know we care, and that we are interested in their world and their problems, not just the narrow segment upon which we are focusing briefly. The DP organization, as a

whole, must have time for users, for their questions, for their complaints. We must provide both an open door and somebody at home to listen when the door is used.

It is also incumbent upon the management of your organization to make sure that the DP/MIS function is represented at the very highest levels of decision-making and corporate power. It is unfortunate, but true, that even though the person in charge of the company's stockroom may be able to come up with some ideas which would revolutionize the way the organization does business, it's unlikely that anyone will listen to him. The DP/MIS function should possess visible corporate clout at all management levels. The user's perception of our capabilities and ability to change things for the better is at least as important as reality.

This also means that we cannot expect to lock ourselves in our rubber-tiled towers and patiently await suitable recognition from the rest of the company of our innate genius. Regardless of the undoubted quality of our systems and DP solutions, they stand a good chance of going unrecognized and unappreciated unless we tell people about them. Each application, each accomplishment, each proposal must be sold to our users. This does not mean that we can go around blowing our own horn incessently as a substitute for real accomplishment. It just means that few people are likely to recognize your worth intuitively, or appreciate your ideas for a new project, unless you take the time to demonstrate it for them. The more a user becomes a believer in a project or a procedure, the more likely that user will help you bring off a success story, rather than a partial fix to a problem or a disaster. A user who is enthusiastic about your project and your operation will perform infinitely better than one who is going along with you because he has been told to. When Machiavelli stated that nothing is as doubtful of success as the creation of a new system, he was specifically referring to the tendency of people to be at best lukewarm toward change, and at worst unalterably opposed.[2]

Leslie Matthies identifies four possible attitudes of users toward a planned new system, in addition to overt opposition:

1. The new system is a threat to personal status, income, job security, personal importance, prestige. The people may passively work for its failure. (Quiet sabotage)

2. They do not think it is any better. Disdain for the new system. Hope it fails. (Passive sabotage)

3. Un-enthusiastic acceptance. Shrugged shoulders. "Just another change."

4. Enthusiastic acceptance. "When are you going to get it going?" "Let us get on with it."[3]

Obviously, we must direct our efforts toward eliciting response number four from our users. To do this, we must pay some attention to the factors which can motivate the user toward cooperation with us, with his fellow workers, and with the system. Matthies lists five of these motivators:

1. He is in on what is going on.

2. He has personal identification. He is not just a cog in the wheel.

3. He gets personal satisfaction from his work.

4. When he accomplishes something, someone that he respects expresses appreciation for his accomplishment.

5. As a worker, he has the respect of his fellow workers and of his supervisor.[4]

There is no surer way to bring about minimal cooperation from a worker than for management to stroll over to his workstation and tell him that a new system (about which he knew nothing) will be installed in the near future, and present him with a new list of procedures. Almost as bad is the typically arrogant announcement that a new system is about to be designed or that a systems analyst or programmer/analyst will be coming around in the near future to decide if there is a better way to do the job than currently is the case. Don't surprise a user with something brand-new, something that you can tell him is for his own good. You will receive exactly the cooperation you deserve.

People want to feel that they have some control over their own destiny, over what they are doing. This has implications at all levels of system design, down to some of the programming specifics we will cover later. In the design phase of a project we need concentrate on only two facts:

1. The user's need for acceptance, for status, for the recognition/respect of his fellow workers, demands that he be consulted if we want him to cooperate with us.

2. Unless we work directly with the actual users of the proposed system, our planning is doomed to failure. One can't depend on supervisors and managers for information about the task characteristics of all the people in their departments, and one certainly can't expect to pick up the knowledge accrued by a worker over the last 10 years from a formal job description.

I am a firm believer in the concept that the net time needed to produce a successful system is inversely related to the amount of time you spend with the people actually doing the job...or even doing the job yourself, when feasible.

Actively consult the users; have them come up with elements of the solution that they can call their own. Even if you have managed to determine the form of a much-needed revision to current procedures, don't be the one to suggest it. Guide the user to that point. You'll get credit enough if the new system actually works, and it has a much better chance of working if everybody concerned feels he was a part of the solution, and has a vested interest in its success.

After you have asked the question, listen to the answer. Don't fake it; don't tune your mind to another channel while you nod and make encouraging noises as the user talks. Remember, there were two reasons for consulting the user. One is to get him involved. The second is that he has at least as much (probably more) to contribute to a successful system as you do.

We can now consider the actual process of system design. As is obvious from what I have said above, it is essential that the user be involved in the system design from its inception.* Each step in the design process should be undertaken only with the active involvement of your users, and with their blessing.

This involvement must be real. Any inference the user can draw from your attitude or bearing that his presence in the process is pro forma is fatal. Every step of the way, you must guard against the arrogance that you are in possession of the "big picture," and know the user's place in it better than he does. This attitude will be sensed quite rapidly, and cooperation will fade away. Integrating the efforts of the data processing department and user department into a coherent process of system design is not easy. James Martin likens the traditional method to another historically inept procedure:

> It is rather like a Victorian missionary first entering an African village. Unlike the missionary, they have to

*Throughout this paper I am using USER as a generic term. Male gender is used in pronouns for simplicity's sake. The term user, depending on the context, refers to data entry clerks, mid-level managers, executives, or all simultaneously. I am assuming that the reader can always supply the appropriate user level to the discussion, since it is incumbent upon us as data processing professionals to do just that.

> produce a very precise document - the specification of requirements.
>
> The missionary is steeped in computer terminology and analysis methods. The villagers' culture is accounting, plans and budgets, or production control - cultures with a complex folklore. They use different languages. Somehow they are supposed to communicate with no ambiguities or misunderstandings. If the missionary is skillful at communicating and can offer the villagers a promise of better things, they can begin to learn each other's conceptual framework. However, there is no way that either can understand the nuances and subtleties of the other's way of thinking.[5]

Somehow, therefore, we must find a way of better integrating the efforts of the system user and the MIS "designer." This attitude is somewhat alien to American businesses, which have traditionally imposed new structures and methods of operation on their workers without a great deal of consultation. We have always been "top-down" in this respect. The Japanese, on the other hand, have emphasized a bottom-up approach to decision-making. By the time an organizational change occurs, all parties likely to be affected by the decision are drawn into the process, and made to feel that their participation is essential to the process. Enough time is budgeted for the decision-making procedures to allow effective input by all concerned.[6]

I am not advocating a retreat from the responsibilities of leadership or abandoning all aspects of our American form of management. It is simply necessary to move toward a more participatory style of system design. Such a process has infinitely greater chances for success than <u>any</u> type of imposed structure, no matter how well thought out or logical it may be.

Make sure that the user knows that you appreciate and value his participation. Again, be careful to avoid any false notes here. Praise the user's involvement and ideas only when appropriate. Don't issue a general "Thanks for your help," and let it go at that. Make it specific; e.g., "Your idea for customer classification will really simplify our design." Don't allow the hearer to believe that you are only praising him in order to get him to do something else. Make sure that he can relish the comment without feeling that you are mortgaging his future. These thoughts, as well as additional discussion on the proper use of praise as a motivator may be found in Paul Timm's excellent book on Managerial Communication.[7]

The approach to systems design outlined above doesn't mesh

precisely with today's emphasis on structured analysis and design. Although I find these techniques to be valid procedural tools, certain precautions should be observed in their use:

1. Be careful of anything that purports to impose a structure of our own creation on the user's domain. I understand that this is not the intent of the structured techniques, but there is always the danger that we find order where in fact there is none. As long as the tool is regarded only as a means of making the user's system more comprehensible for analysis, and as long as the user himself can understand what we're doing with his comments and opinions, it can be helpful.

2. Don't rely on formal systems design documents as a means of communication between you and your users. This kind of document has become a traditional means of covering your vulnerable parts by requiring that users sign off on it at the close of each step of system development. Unfortunately, it often leaves us no closer to a true understanding of what will or should be done. Again, I borrow from James Martin:

> The specification document usually has the following unfortunate characteristics. It is so long that key managers do not read it at all. They read the summary. It is incredibly boring and this causes it to be skip-read by many people who should read it fully. It contains technical terms, systems analysis charts, and various forms of professional shorthand, all of which the end users do not fully understand. It contains words which have very precise meanings in the user areas, but which programmers do not understand, and most systems analysts do not appreciate their nuances.[8]

3. When you measure something, you change what you measure. The act of observation places constraints on behavior which are not usually there. Recognize this, and recognize the fact that a system built on formal procedures which ignores the informal procedures that have gradually grown up and supplanted the original structure, is doomed to failure.

4. Perhaps most important is the fact that structured analysis, as it is most frequently practiced, is still only a tool for documenting a system as it is currently being operated. It is easy to overlook the fact that interactive systems, once designed and implemented, tend to change rapidly (assuming any leeway in the system at all). Making terminals available to users changes the game being played, not just the rules of the game.

Given these caveats, I believe there is still a place for the techniques of structured analysis and design. Used properly, these tools can contribute significantly to the process of specification and design of a certain type of system. The alternative to these procedures is what Martin called "User-Driven Computing." As the power of our software tools increases, especially packages which are meant to be used by end users to generate applications, more and more of the responsibility and capability for application development is being shifted to non-DP personnel. Martin's superb treatment of this evolution (revolution?), and the place of data processing personnel is the topic of the book I referenced twice above.

Although I have emphasized that the end user must be involved in the system design process from the beginning of the project until its conclusion, I have thus far concentrated solely on the beginning of the process: requirements analysis and system specification. The User must walk hand in hand with the MIS department through the various checkpoints that follow preliminary agreement on what the system is supposed to do. One particularly important place for the user to be involved is in system and module walk-throughs.

Walk-throughs are a necessary step in every level of system development. My staff walks through system narratives, program narratives, program code, testing procedures, and user documentation. It sometimes seems like an interminably lengthy process. However, the number of significant logic or specification errors discovered after implementation is minimal - and those are the bugs that really hurt.

The user belongs in walk-throughs. Common sense is important here. It must be the right user and the right walk-through. No user belongs in a code walk-through. A manager should not be substituted where an entry clerk is appropriate, although there are levels at which input from each is needed. Therefore, make sure your walk-throughs are structured appropriately (read Freedman and Weinberg's book on the subject[9]), but neglect this step at your own risk.

Desirable System Characteristics

Whatever the methods we use for systems analysis and design, our primary goal is to come up with systems that work. We will now consider some of the characteristics of workable systems, emphasizing, of course, those built around the "people" aspects of the system.

We can begin by borrowing two thoughts from John Gall's treatise on Systemantics. The first is his Systems Law of Gravity (alternatively called the Vector Theory of Systems).

Systems run best when designed to run downhill. Gall explains:

> In human terms, this means working with human tendencies rather than against them. For example, a state-run lottery flourishes even in times of economic depression because its function is aligned with the basic human instinct to gamble a small stake in hopes of a large reward.[10]

Recognize the role that self interest plays in system performance. For an employee to consistently contribute to a system's data base, he must receive something in return. This implies that you can't build a management information system on the backs of the little people. Unless users at all levels of the system have a vested interest in its accuracy, timeliness, and general performance, you are going to suffer breakdowns. Few days go by without reminding me of this truth. One particularly glaring example lies in a system which depended, for a single piece of critical data, on the performance of individuals in a department otherwise unrelated to the system. The folks who were to provide this data received nothing from the system, and regarded their notation of this item on their paperwork as unessential to their function. Since it sometimes required a significant effort to come up with the item, and since they were receiving no regular feedback about their performance of the task, compliance with the requirement became non-existent when things got rushed, and wasn't much better under normal circumstances.

Even more important is Gall's next theory. The ramifications of this simple statement extend into almost every aspect of system design.

Loose systems last longer and function better. Gall comments:

> ...don't make the system too tight. This is usually done in the name of efficiency, or (paradoxically) in the hope of making the system more permanent. Neither goal is achieved if the resulting system (a) doesn't work at all; (b) disintegrates; or (c) rapidly loses steam and peters out.[11]

Keep the user's job interesting. Give him some leeway. Our so-called sophistication permits us to reduce many data input jobs to rote procedures. DON'T! It's almost always a mistake. Let the employee do something, use some discretion. Give him the feedback and accountability to ensure that the tools are available for him to approach perfection. Don't try to approach that blissful state

without him. Mindless tasks will be performed in an appropriately mindless manner, and I defy you to design around that phenomenon. Keep the data entry operator's mind active and interested.

It is not necessary to design computerized systems in such a manner that users of the system are doomed to dull, uninteresting work. The principles of job enrichment listed below are applicable in data entry situations.

> The program should make it possible for the same employee to handle a wide variety of transaction types.
>
> A single employee should be able to complete a transaction without the need to pass it on to another employee.
>
> Computer produced outputs should allow for the identification of the employee who made the transaction appear.
>
> Build into the job as many decision and problem-solving activities which affect the service and quality levels of the work as possible. Examples are: setting work priorities, resolving complaints and mistakes, sending work back for more information, direct communication with other areas.
>
> Provide opportunity for frequent and direct information on how well the job is being performed.[12]

Characteristics of INTERACTIVE Systems

The goal of interactive system design can be related to the following quote referenced in Ben Schneiderman's text on Software Psychology:

> If the point of contact between the product and the people becomes a point of friction, then the industrial designer has failed. If on the other hand people are made safer, more comfortable, more eager to purchase, more efficient - or just plain happier - the designer has succeeded.[13]

We must be concerned with every aspect of the system from sign on to sign off. As usual, the finest summary of characteristics to strive for is provided by James Martin. He is speaking specifically of desirable properties of user dialogues.

> The means of establishing contact with the computer and signing on should be simple natural and obvious (with appropriate security routines).
>
> The user should be required to know as little as possible in order to get started.
>
> The dialogue should completely avoid forcing the user to remember mnemonics.

> The dialogue should completely avoid forcing the user to remember formats or entry sequences.
>
> The dialogue should never put the user in a situation where he does not know what to do next.
>
> The dialogue should provide a simple, natural, and obvious means for the user to recover from any mistakes or surprises. A good facility is that of back tracking to the point before the surprise occurred.
>
> The response times should be fast enough to avoid frustrating the user.
>
> The software should have good HELP functions with which any facility can be explained.
>
> The software should be self-teaching, with computer-aided instruction at the same terminal.
>
> It should be human-factored well enough that the end user for whom it is intended can become skilled with it and obtain valuable results after a 2-day training course.[14]

Again, the emphasis is the same. Let the user feel IN CONTROL. Let him feel like he knows what is going on. It is so important that the person sitting at the terminal feel involved, rather than like a peripheral device. When the enter key is pressed on a block mode screen, don't let the user wonder what's happening while you go off to do your data base lookups. Software designers are getting somewhat better about this than they used to be. It is now common for the equivalent of "I'm working on your request" to be returned to the user, but even this lacks effectiveness as seconds turn into minutes. Some measure of progress should be given whenever any lengthy delay is encountered. The progress reports given while Adager is performing data set transformations are a simple but effective example of this.

Similarly, Jim May provided us with a discussion of several techniques for improving the user's perception of the system's performance in his widely reproduced article, "Programming for Performance."[15]

The amount of finagling you can do with the user's impression of response time is, of course, finite. Much thought must be given to the design of data bases planned for interactive update and inquiry. It is unfortunate, but true, that some of the same features which can optimize a data base for on-line inquiry (e.g. sorted chains) can make

the addition or deletion of records subject to intolerable delays. We have fallen prey to the desire to make on-line inquiry as instantaneous as possible in our installation from time to time. In one instance, the delay associated with adding new entries to some product chains became so pronounced that our inventory entry clerks could have gotten an adequate night's sleep between successful transactions. Thus, despite better than adequate response times for on-line inquiries, I was forced to throw in the towel, and change the primary entry into several data sets. It was particularly aggravating since the solution chosen had been suggested to me at the time the system was being designed, and I had rejected it. In retrospect, it is difficult for me to understand why I would have nothing to do with the suggestion. We constructed a concatinated key out of five elements present in all entries in the affected sets. The chains into the largest data sets were reduced to negligible length by this means; hence a consequent reduction in the time it took to add and delete entries was immediately noticeable. All the work is done by the CPU in a manner which is completely transparent to the user - he never needs to know anything about the key, and he is infinitely happier with his response time.

It is not always possible to optimize data bases for both update and on-line inquiry. Some users find it necessary to develop data bases which are in fact spin-offs from production data bases, but which are optimized for on-line inquiry. In certain situations, this duplication of data storage is simply unavoidable. On-line inquiry functions are often time-critical. Receiving answers to today's questions tomorrow is meaningless when the manager has moved on to a whole new set of problems. There is no adequate cosmetic solution for a data base which cannot provide reasonable response times.

Let us now return to the type of feedback we provide the interactive user, and in particular to the computer's half of the dialogue. Don't get cutesy! Don't be patronizing. The computer is not a person, and the user is not sitting at the terminal for the privilege of playing word games with it. Almost all of us who program interactively have gone through an adolescent stage wherein we would provide users with responses such as, "Now (user's name)...you can do better than that...Try it again, good buddy." This may be amusing the first time around, but it gets old very fast. Let the user feel in control of the system, not controlled by it. From the beginning of the design process through the final stages of acceptance testing, and then as long as the system is operational, the interactive user should never have to feel that he is being man-handled

by something over which he has no control.

> A driving force in human behavior is the desire to control. Some individuals have powerful needs to attain and maintain control of their total environment; others are less strongly motivated in this direction and are more accepting of their fate. With respect to using computers, the desire for control apparently increases with experience. Novice terminal users and children are perfectly willing to follow the computer's instructions and accept the computer as the controlling agent in the interaction. With experience and maturity, users resent the computer's dominance, and prefer to use the computer as a tool. These users perceive the computer as merely an aid in accomplishing their job or personal objectives and resent messages which suggest the computer is in charge.[16]

There has been a surprising amount of research done on the design of interactive systems. Space doesn't permit me to summarize even the most significant studies in this paper. However, some findings were universal to almost all of the research efforts, and are summarized below. I refer you to Schneiderman's chapter on "Designing Interactive Systems" in Software Psychology.

The system should be tailorable to the level of expertise displayed by the user. Ideally, the system should be able to determine this through the dialogue process and retain this knowledge from session to session. Minimally, the system must deal in different ways with users of different sophistication.

The user should be able to learn the system as he uses it; the amount he has to learn before he begins use of the system should be minimized. As the user grows in sophistication, he should be able to skip over or avoid items provided for the novice. While it is helpful for a beginner to be able to step his way through various levels of menus, it can become time-consuming and annoying for the individual who knows exactly where he wants to go. Therefore, the system should provide shortcuts for him. HP's new financial system is an excellent example of this principle. The user can step through screen after screen by pressing appropriate function keys, or can enter the name of the screen he wants to use, thereby skipping interim steps.

The HELP facility is critical to the operation of interactive systems. It must provide an appropriate level of assistance to the user, without being obtrusive. Smith, Dennis & Gaylord, in the character mode screens of their various systems, provide a field by field description/HELP

facility which can be disabled by the user when he feels he has become an "expert." He can reactivate the facility for a field simply by typing a question mark as his response to that field. In HP's new financial system, an extensive HELP facility is but a function key away. Both of these systems provide the very important feature of customization for HELP messages - you can tailor the messages to your own installation.

The system should be user-modifiable, allowing a personalized version of the system to automatically interface with the code provided without source code modifications.

The system must respond consistently to user inputs (nothing is more infuriating than something that works most of the time), and must provide a response to every possible user input.

The need for memorization of codes and mnemonics should be minimal. Keep it simple!

Error messages, of course, should give the user enough information to know what to do next. General messages, which apply to more than one screen or situation should be avoided if possible (I've yet to find a situation where the error message can't be tailored to the specific problem the user is facing). Returning "Invalid Key" or "Invalid Entry" to the user usually is both unhelpful and annoying.

The system must be forgiving. Allow those errors to be repaired. Ask the user if he really wants to do a particularly dangerous entry (sometimes you can itemize the consequences to avoid "I didn't know that would happen!"). Allow the user to back out of sticky situations when he changes his mind. Dialog should be interruptable (to be either resumed or backed out of).

Don't ever let the user wonder if his entry was successful. If the data base was successfully updated, tell him so. If a transaction was only partially completed, let him know exactly where he stands. Better yet, don't ever let logically connected transactions be entered only partially.

In other words, let the user be in control; let him procede at his own pace, in his own way.

Data Entry Screens

Another aspect of interactive systems to which we must devote some attention is the design of Data Entry Screens. V-Plus makes it rather easy to design screens. More often than seems reasonable, however, we give too little thought

to the design. This is unfortunate, since the appearance and layout of a system's data entry screens are the initial point of contact between a user and the system, and can be responsible for lasting impressions.

Actually, the primary design considerations for data entry screens are relatively few in number:

1) The screen should present a good appearance. Some may question why I place this item first. Just as I have long been convinced that the appearance of a hardcopy report has as much to do with its acceptance by users as the quality of the data it contains (and almost as much as its timeliness), I am certain that, if your screens are attractive, you have cleared the most important hurdle in screen design. Above all, avoid clutter and "busyness."

2) Standardize your screen layouts. They should have similar appearances throughout your system, and, for that matter, throughout your installation. Let data entry operators know where to look for certain items, both for entry and display.

3) If there is an input document from which the data entry operator is working, the screen layout should correspond as closely as possible to that document. Hopping from place to place on the original document is confusing, time-consuming, and error-prone.

   Avoid using the data entry screen as a keypunch machine. If you are designing a data entry system, the input document and the CRT screen should be designed <u>at the same time</u>, so that both data gathering and data input can be optimized. There is rarely an excuse for just accepting the old input form without critically examining it. Remember, you are designing a data entry <u>system</u>.

4) Within the constraints of items 1 through 3 above, design the screen for speed and convenience of input. Group the most frequently entered items so that the operator does not have to tab through less used fields to get to them. Remember, this is also the characteristic of a good data gathering instrument. Design the input form and screen together.

5) Label fields clearly and consistently. Make sure, to the extent possible, that the operator possesses the information needed to use the screen <u>on</u> the screen, or at the touch of a HELP key. If a HELP function key is used, return the operator to the place he left.

A Word about User Documentation

What percentage of currently operating systems provide their users with adequate documentation? If such documentation is provided, are updates regularly supplied? If the updates are supplied, does the user actually take the time to keep his copy of the manual up to date?

You and I both know that by the time we have answered question #3 above, we are working with very small percentages. Both the user and the analyst have little interest in documentation until it is needed and found to be missing. This situation exists despite the fact that all of us in the data processing business are admonished with nauseating regularity to clean up our act in this area.

At this juncture, I see no point in adding my voice to the legions calling for clear, up-to-date user documentation. Nobody would listen anyway. However, I would like you to consider three points:

1) The user has to learn the system. Something has to teach the user the system, and the previous operator or person who filled that position may no longer be available. Therefore, the system should be able to provide the necessary documentation to get the operator started. That is, the documentation should be online, not in a manual which hasn't been opened or updated since the last operator learned the job.

2) The two types of online documentation needed for the user are:

   a) Something to get him started; a walk-through or demo

   b) An extensive, easily accessible HELP facility to answer the large majority of questions that come up in the day to day operation of the system. The data here should be easily modifiable and customizable.

3) Yes there must be some written documentation. There are some things that simply can't be fitted on a screen in a coherent manner. This is why computer manufacturer's provide reference manuals to their users, and that is what needs to be given to a user of any system we are providing them. These are not cookbooks, not how-to manuals. The reference manual should contain all the information the user is ever likely to need. Most important, of all the features the document should have, is the necessity for extensive indexing. Don't put him through the agony of knowing the answer is in there somewhere, but not knowing where to find it.

Each of the topics I have touched on above could be expanded upon. In many cases, I have tried to provide a reference or two to a particularly useful publication on a given topic.

There are other topics I have not even addressed. Design of output reports, installation procedures, training techniques all must be considered when you are trying to build systems which users are going to be pleased with, and actually use. We have not considered the growing Information Center phenomenon, a valuable technique for helping users to help themselves. And all of the above is for naught if our operations department is unable to provide data in a timely fashion or the personnel on the MIS staff, at all levels, are anything but cooperative and pleasant for the user to work with.

Thus we return to the central thesis. Designing "user-friendly" systems is dependent upon a state of mind, an attitude. The MIS staff, the computer operator, the systems analyst should all possess one common thought: The only reason their jobs exist is to help the users of the system. I for one think it advisable to please the people responsible for the existence of your job, even if it means "pampering" them. Amazingly, the benefits derived from that attitude far outweigh the effort involved.

References

1. Robert Townsend, Up the Organization (New York: Alfred A. Knopf, Inc., 1970), p. 36.

2. Machiavelli, The Prince, 1513

3. Leslie Matthies, "How Important are People," Peopleware in Systems (Cleveland: Association for Systems Management, 1976), p. 2.

4. Ibid., page 3.

5. James Martin, Application Development Without Programmers (Englewood Cliffs, N.J., Prentice-Hall, 1982), p. 59.

6. John M. Nichols, "Systems/User Lessons from the Japanese," Journal of Systems Management, September 1982.

7. Paul R. Timm, Managerial Communication (Englewood Cliffs, N.J.: Prentice-Hall, 1980), p 135.

8. Martin, page 60.

9. Daniel P. Freedman and Gerald M. Weinberg, Handbook of Walkthroughs, Inspections, and Technical Reviews (Boston: Little, Brown and Company, 1982).

10. John Gall, Systemantics: How Systems Work and Especially How They Fail (New York: Quadrangle/New York Times, 1975), p. 70.

11. Ibid.

12. Lyle Yorks, "Job Enrichment Boosts Performance," Peopleware in Systems, pp. 60-61.

13. Henry Dreyfuss, Designing for People (1955).

14. Martin, page 115.

15. Jim May, "Programming for Performance," Journal of the HP 3000 Users Group, July/December 1982.

16. Ben Shneiderman, Software Psychology: Human Factors in Computer and Information Systems (Cambridge, Mass.: Winthrop Publishers, 1980), p. 226.

A PSYCHOLOGIST LOOKS AT THE DP SHOP

E. R. Simmons, Ph.D.

COLE & VAN SICKLE COMPANY

And what he sees is a rapidly changing, high technology work place with high ability, high achieving independent thinkers working in a high stress, low morale, ambiguous environment; a very fertile field for psychologists.

In my personal contact with DP personnel, I have detected a high level of frustration, disenchantment, unrest and general inner turmoil which appeared to spill over into the shop atmosphere and create a hostile work situation. This condition results in more stress and becomes a vicious circle, stress leading to more frustration and more frustration leading to more stress. The frustration-stress syndrome seemed to be a combination of things--man against machine (how am I going to get this antiquated, obsolete relic to accept, store, process and produce my data in usable form with a minimum time lag?)--labor-management (what will it take to get management to see that I need other tools in order to get the work done?) and interpersonal problems (how am I going to convince my colleagues that their approach is not suitable to the task at hand?).

Usually this set of conditions poses an impossible task and the DP-er resigns and finds another shop and begins the process of developing another hostile environment from which he must remove himself at some point in time. Of course, this is not due entirely to the worker alone. Insensitive company policy, poor managerial strategies and pressures inherent in the DP Business all combine to produce this intolerable situation. The end result is that in addition to the dollars and cents cost to the company and the individual, there is a price that has to be paid by the DP-er in terms of his affected interpersonal relationships, especially with his spouse or significant other person, and his children, if he is a family man, which most are. Besides the cost of replacement of the worker by the company, there is also the fact that the stage is set for the same thing to happen again when a new employee arrives on the scene. He will inherit all of the pressures of the situation plus complications that arise due to having to replace an unhappy employee who has not made a great effort to get his shop in order before his departure.

Repeating this cycle on a continuing basis is not profitable to individuals or companies or the industry at large. It is a situation that begs to be corrected and correction must come from a joint effort of worker and management. The DP-er must make some changes as an individual and management must begin to look more at people in their organizations and their needs.

Ann Work[1] looking ahead in 1980, said "It seems to me that major hardware design considerations were established in the '60's and major software considerations in the '70's. The focus for the next decade is people; how we manage people, what we expect of people within data processing, and what we expect of people in the user community." That observation appears to be consistent with what might be expected in a logical progression of events and it is certainly a statement with which hardly anyone could find fault, if for no other reason than that DP shops were beset with people problems. Something obviously needed to be done. As an example of the gravity of the situation, one manager[2] is quoted as saying "Programmers have pretty well been the masters of their own destinies throughout the decade. They have been well paid; they have named their hours of work; they have hopped from job to job with impunity, leaving behind deskfuls of chaos; they have been wined and dined, coddled and humored, promoted and pampered. If ever a group of employees rose to Cinderelladom overnight, the computer specialists were that group. They were immune to control; they were immune to discipline; they were immune to challenge; they were immune to competition". Granting that this statement may tell more about the manager and his approach to handling people than it does about the DP-ers themselves, his opinion is not far off the mark when compared with others of the same time period. It was apparent that something needed to be done about conditions that prevailed in the DP shop. Companies and individuals were suffering. Ms. Work was right. It was time to begin to emphasize people in the computer world. Something happened, though, to make it even more difficult than it had been while standards were being set for hardware and software, something that made it more difficult than ever to take the time and effort to concentrate on people. An explosion took place. An information explosion.

In January, 1982, Janet Crane[3] asked "Where is the industry going?" Her answer--"Everywhere at once. The driving force is still technology, and observers are predicting another 10 years' innovation equal to that of the last 25. Data communications. Office automation. CAD/CAM. Ever lower costs for undreamed of power".

In January, 1983, Robert Batt[4] said "The likelihood that large corporations will witness an unprecedented display of computing power at all levels this year is sending management information systems (MIS) managers scurrying to come up with systems planning procedures for coordinating this information explosion".

In the kind of milieu portrayed in the above statements, it would be difficult to envision much being done about people problems in the next few years, as suggested by Ms. Work. However, the very fact of this communications explosion makes it more imperative than ever that steps be taken to get the most out of the manpower available. If this area isn't addressed by both companies and individuals, the public, as well as the industry will pay a high price for the industry's failure to take care of its business in a reasonable manner. As mentioned in the beginning of this article, the DP shop is an extremely stressful place and without dedicated, purposeful planning, it is going to get worse.

Stress is the key factor in burnout and turnover of manpower in any industry, business or profession, and the DP industry can ill afford to lose people from its ranks, whether to other shops (because this is costly) or to other industries or professions. Both the individuals and the companies have a stake in this endeavor and both share the responsibility for making something happen to turn the tide.

Let us first look at the professional DP person. Studies have identified a distinct psychological profile for DP-ers. All investigators have come up with a conclusion that he, or she (there is little difference between males and females in terms of their psychological profile) has a very low social need, low needs for social recognition, is unsociable, not impressed by social conventions that others respect and finally, have low social need strength. What we seem to have here is a person who would be better off in a monastery (one DP-er, at least, has chosen that option) or, at least, on some desert isle. Now, I must say that in my contact with DP people, I have not found them to be all that antisocial or asocial. As a group, they seem to be more than willing to talk about their work, their families, politics, the condition of the country, the company they work for and yes, I have even heard some of them make comments about sex. How, then, do we explain this apparent discrepancy in observation and objective measurement of DP-ers? It really isn't too difficult if one understands the nature of psycho-

logical instruments used to determine these sorts of things. First, we set up a few criteria by which we determine a person's sociability; for example, he seeks out the company of others; he desires the approval of others and so on. Then, we develop a plan of action to determine whether these characteristics can be found in an individual. So, we set up a battery of questions; for example, would you rather read a book or go to a party? There is an obvious answer here for one who is to be considered sociable. Then, there are other questions of the same vein. Now, let us say that the person who says "I had rather read a book" is classified as an unsocial person. (In all due fairness to my own profession, let me say that this is very much oversimplified and that psychologists do use personality assessment instruments to great advantage in assisting people who find themselves desirous of making changes in their behavior). Now, this unsocial person happens to be a DP-er and it is found by further investigation that the great majority of DP-ers are unsocial, using the same criteria. What appears to be happening here is that DP people do not manifest their social needs in commonly accepted ways; therefore they develop an image. Human beings, having the characteristics they do, react to others in ways that are dictated by their perception of them. In other words, if this person is unsocial, I am certainly not going to force myself upon him. If he wants to socialize with me, I want him to make some overtures to that effect. If this is the attitude taken by an individual, he cannot expect a funfilled evening with others in whose company he finds himself, whether they be DP-ers or morticians, or whatever. The fact of the matter is that DP-ers as a group are brighter than the rest of us, intellectually, that is. They do spend a lot of time reading for information and understanding of complicated matters and they live in a very esoteric world to which the rest of us are not privy. Besides that, they speak of strange and frightening things, like nanoseconds and picoseconds and calculations that take a thousandth of a billionth of a second. Now, the rest of us know that such things really cannot be, but the DP-ers seem to think so, so who are we to disturb them. And everybody knows they are unsocial people, so why bother to try to understand their world or try to get them to understand ours? In other words, what DP-ers have is an image problem. And it could stand some modification; however, to this observer, it seems that they cherish it and stubbornly cling to it because, even though it works to their disadvantage on occasion, it is theirs and sets them apart and helps maintain a camaraderie that has been noticed by more than one observer. For, in the literature describing DP-ers, almost all agree that they are

more loyal to their profession than to the companies that employ them. To that, I ask, what professional isn't? Doctors and lawyers even go so far as to set up ethical standards and bind their colleagues to upholding them and practicing them. Nothing is more important than having the respect of your colleagues as far as a professional is concerned. Even Psychologists have this infirmity.

Since there is nothing really wrong with the image the DP-ers present to the world, should they be concerned with it? The answer is a resounding YES. For the simple reason that it interferes with getting their work done, with establishing rapport with others and with getting them the recognition they deserve. It is not that recognition is important for self-aggrandizement, but it enables them to have input in decisions that affect the shops they labor in for their good and the good of the companies they represent. DP-ers simply must become more PR (Public Relations) conscious. With the rapidly changing DP environment, the DP-er is going to find himself in a position of Consultant more than that of systems analyst and programmer. As one person[5] expresses it, "The new technologies, methodologies and demands are turning what was an exotic art into a more civilized science. Effective user relations, not elegant coding, is what's important". The wise DP-er will take heed and respond accordingly.

In addition to developing PR skills, the DP-er should learn something about the nature of stress and its implications. I am sure that most DP-ers do not realize the nature and magnitude of the stress they endure from day to day in their shops; or, if they do, they do not express it. Somehow, this fits into their profile as persons of stoic acceptance of what life has to offer, or self-discipline that enables them to bear up under any strain. The most damaging aspect of DP shop stress is that it never goes away, not even when the work "day" is over. This is not true for most workers; for example, Air Traffic Controllers are subjected to highly stressful situations in their work, but when the workday is over, they are relieved of the situation unless they have made a mistake and endangered somebody's life and know they are going to have to answer for it. That does not happen every day because, if it does, they will soon find themselves in the market for another job. The DP-er, on the other hand, never has a moment free from concern about his program or his system or his computer. He even wonders if his sleep will be disturbed by a call saying that the computer is down. In addition, he is almost always working on a deadline and if the work isn't done, there will be complaints from several sources. He rarely enjoys the

luxury of devoting himself completely to a clearly outlined, unambiguous task, seeing it through to its completion, without interruption and presenting it to the boss, who is highly elated at such a job well done and with such speed. No; his day is filled with interruptions, new directives from on high about budgetary considerations, other high priority tasks that require the use of the terminal and his energy, computers that are contrary and inherited programs that are worse. It gets to be a real chore just to exist from day to day, patching here and patching there and getting this bit of information for the sales department and that bit for the personnel group and another for upper management, all of whom seem to be more unhappy than pleased.

Other sources of stress are the decentralization of DP and the continuing advances in technology. The former is taking place on such a large scale that no one can really keep up with it and every new proliferation brings with it new problems. The entire field is becoming user-oriented. Users who know nothing of DP are now dictating, in a sense, what will be done. They have at their hands personal computers and want to control their own destinies, so to speak. They tell the DP people what they want with no idea of what that might entail in terms of money, time and manpower, to say nothing of technology.

As far as continuing advances in technology, both management and non-management are hard at work simply trying to keep up and maintain their status. It is quite easy for the worker as well as the machine to become obsolete. One has to keep up with the new wave in technology and this, coupled with their daily responsibilities in the workplace, imposes a great burden on DP professionals.

Coupled with all of the stress of the workplace is its effect on the home front, which compounds and intensifies the damage to the individual. One DP professional informed me of a situation that developed in a large corporation with which he had been associated. Ten professionals were involved in a two-year project that so consumed their time and energies that they had little left of either for the home. At the end of the two years, eight of the DP-ers were divorced. One could argue that conditions at home drove the professionals to spend more time with the project than was really necessary just to get away from a bad home situation, but eight out of ten is a rather large percentage even for this day and time.

So, where does it all lead? What does it mean? If stress is bad and there is a great deal of it, what could and should be done? If the DP image needs a facelift, how is it accomplished. Where do we go from here? Let us consider.

First, we will look again at the personal characteristics of DP professionals. It has been noted that they are not considered to be very sociable (although this is certainly in question), that they are somewhat stoic and exercise a great degree of self-discipline. Other characteristics identified are the ability to do abstract thinking, use cognitive structure in problem solving rather than intuition, low need for aggression and personal conflict. They also are extremely conscientious, responsible and persistent. Strength of character and a concern with moral standards and values also describe the DP professional. We can see that, in spite of the fact that they are not considered to be very sociable beings, these are people we would like to know and have for friends. They are people one could depend on, people who would be loyal and supportive. It is my suggestion that with these strengths, it should not be too great a task for the DP-er to do the things that would make for better relations with others, including the company for which he works.

It seems to me that there are two essential things the DP person must respond to. One is his image and the other is his reaction to stress. As we have pointed out, his image is one of aloofness, of being a loner, noncommunicative, sufficient unto himself. In regard to stress, it seems that he behaves as though he is unaffected by it, which is not so. In order to accomplish the most good in his chosen field and reach his potential personally, he will need to do the following things:

(1) Realize that he and his profession are negatively viewed by society as a whole. Society simply does not like to be computerized. We do not like to become numbers to be manipulated and controlled. Therefore, we regard professionals in the field as a personal threat. This feeling isn't verbalized, usually, but it is there, nonetheless.

(2) Make an effort to do some PR for his profession. Do something to counteract the negative feeling of society. Now, about all anyone hears about computers is that they are "down" or they "fouled up my account". People need to have more information about the great strides made in Medicine, Communications, Research, Transportation, etc. because of

computers. Of course, this is a job for Madison Avenue, but each DP professional should be less reticent and more vocal in support of his industry.

(3) Make an effort to be more communicative with the uninitiated, recognizing that most of us really have little knowledge or understanding of the field.

(4) Prepare to get away from the computer more and serve in a Consultant capacity to the end user.

(5) Learn to compromise and make allowances for the needs of Business as a whole.

(6) Accept the fact that businesses must be run according to sound Business principles and do not parade your nonconformity except in appropriate circumstances.

(7) Accept the fact that you can be affected by stress and learn satisfactory ways of coping with it, rather than assuming you are immune to it.

All of these recommendations are simple and relatively easy to do, but they will probably require some professional help in certain cases. Be sure to seek it when you need it. The very existence of counseling is rooted in the belief that people can change. It is not wise to follow the dictum that "all redheads have hot tempers" or "that's just the way I am - the Psychologist said so". If change is indicated, change.

BUSINESS

Now, let us look at the management or corporate side of the coin. In this day and age, any Business of any size will be using DP personnel. They have a vested interest in keeping their DP professionals on staff and productive. They cannot afford to lose them to stress, burnout and turnover. The identifiable costs in dollars and cents are great, as has been mentioned, and it is really impossible to measure the hidden costs that go with personnel changes in the DP

shop. What does Business do about this?

It appears from an observer's standpoint that Business in general has not figured out what to do with and about DP. It should be perfectly obvious by this time that DP belongs in the mainstream of Business decision making. It has long since ascended from its early function of providing information for the accounting department. It is vital to all phases of Business and should have input on an equal basis with Sales and Marketing, Personnel, etc. The problem is how to accomplish this with people who appear to be somewhat unimpressed by Business generally and seem to prefer to stay in their shops and do their debugging. This, along with technological advances, has made the assimilation difficult. Decentralization, user oriented approaches, sophisticated software and contract professionals along with smarter computers are all at work to solve DP problems, but judging from the observable activity and reported data, Ms. Work's suggestion that we need to work with people holds out the most hope. In reviewing the literature, it becomes apparent that investing time, money and effort in people is what pays off.

James Matteoni[6], Vice President for Levi Strauss, San Francisco is quoted as saying "I think the more you give people, the more they're going to give you. Some take advantage, but good people stay and attract more". During the three-year period from 1977-80, the turnover in his company dropped from 40% down to 18%. During this same period, training funds for employees went from $25,000 in 1977 to $280,000 in 1980. Obviously, a company with that philosophy is going to be doing some other constructive things, but apparently they see a connection.

A different sort of experience led John Bowyer[7], International Harvester Vice President to the conclusion that people were a good investment. After 24 years service, he saw his company descend to a survival level in 1982, coping with, among other things, a 70% annual turnover rate in DP. He had seen good times and bad times. When asked what he had learned during the difficult times, he said "I've learned the value of people who leave when you never expected them to. I've learned that growth and expansion, even in good times, has to be tempered with good planning. And I've learned how important it is to maintain good communication with people. That's what I have learned".

Writers and Consultants agree with Business tycoons.

Merrill Cherlin[8] writes in Datamation that "The best way to prevent burnout may be to work for a progressive company that recognizes that people are the most important asset, and nurtures and challenges and rewards this asset".

Robert Alloway,[9] Consultant, is quoted as saying "In these turbulent times, Senior Executives are proving to be much less accessible to their information systems managers than they used to be. The price paid for such inaccessibility is the development of less-relevant systems and the shortsighted gutting of plans for systems that would best prepare corporations for the future".

Janet Crane[10] writes in Datamation "Despite the widespread attitude that computer folks are technically oriented and want to stay that way, indications are that companies with low turnover rates are those that have directed attention to people management, career development for their systems people". In the same article Ms. Crane quotes Darwin John of Scott Paper Company "When DP people are challenged and given the opportunity to participate in the overall business thrust, turnover is almost obliterated".

Finally, in Computer Decisions[11], we find an example of a company showing its concern for people in general and setting an example for its employees by a statement of philosophy. Hewlett-Packard, in 1957 in their statement of objectives said "To honor our obligations to society by being an economic, intellectual and social asset to each nation and each community in which we operate". L. A. Fulgham, a personnel manager for HP, says turnover at HP is very low.

The picture is clearly in focus. Over and over, we see the same theme repeated. Take care of your people and they will take care of you. The companies enjoying the least turnover are the ones who emphasize the value of good people, who reduce stress and resultant burnout in the lives of their employees.

Companies, then, to get the most return on their investment in the DP-er must do the following:

(1) Be alert to indicators that DP personnel are burning out. They are universal when it comes to dissatisfied workers. They take longer lunch hours, they are absent more. They spend a lot of time standing around, visiting, etc.

(2) Have flexible working hours as much as possible. Creativity cannot be turned on at 8 A.M. and off at 5 P.M.

(3) Make provision for educational time for your DP professionals to keep updated.

(4) Make arrangements for comp time. Most DP-ers put in a lot of extra time. They don't necessarily benefit by receiving more money. They really need time off.

(5) Identify the needs of your DP people for motivational purposes.

(6) Make an effort to get the DP-er into the mainstream of the Business operation rather than having him secluded in an isolated area.

(7) Make an effort to structure career paths for your top people rather than put them in a dead end situation early in their careers.

(8) Do not use the same standards to measure DP personnel as you use in other departments.

(9) Learn to appreciate their creative intelligence and use their analytical skills.

(10) Educate your people in company philosophy with a view to developing a sense of loyalty. Give them an opportunity to take pride in the company they represent.

In conclusion, let us say that given the conscientious, moral nature of the majority of professional DP-ers and companies committed to treating them as valued assets, a significant reduction can be made in the stress-burnout-turnover toll that plagues Business and Industry.

BIBLIOGRAPHY

1 Work, Ann
People in DP: The 1980s
Computer Personnel, Vol. 8, No. 3, October 1980, p. 7.

2 Snyders, Jan and Martin Lasden
Managing Programmers To Work Harder And Happier
Computer Decisions, Vol. 2, No. 10, October 1980, p. 46.

3 Crane, Janet
The Changing Role Of The DP Manager
Datamation, Vol. 28, No. 1, January 1982, p. 98.

4 Batt, Robert
MIS Managers Say DP Planning A New Game
Computerworld, Vol. 17, No. 3, January 17, 1983, p. 17.

5 Lasden, Martin
Recycling Displaced Dissatisfied Pros
Computer Decisions, Vol. 12, No. 11, November 1980, p. 48.

6 Ibid.

7 Lasden, Martin
Surviving Corporate Crisis
Computer Decisions, Vol. 15, No. 1, January 1983, p. 88.

8 Cherlin, Merrill
Burnout:Victims And Avoidances
Datamation, Vol. 27, No. 7, July 1981, p. 99.

9 Lasden, Martin
Surviving Corporate Crisis
Computer Decisions, Vol. 15, No. 1, January 1983, p. 90.

10 Crane, op.cit., p. 108.

11 Lasden, Martin
Cutting Turnover With A Japanese Pattern
Computer Decisions, Vol. 13, No. 9, September 1981, p. 148.

State of the Art Human Factors in User Friendly Systems

Wanda Smith
Hewlett Packard

A user friendly system is one in which the design of the interaction between the users and the equipment is enhanced or optimized. This includes both hardware and software and encompasses the environmental aspects in which the interface takes place. The design of the hardware addresses items such as: furniture design, noise and/or heat generated from equipment, adjustability and dimensions, spacing and location of control keys, and character quality on a display screen. Software design features include: formatting, utilization of coding techniques, semantics (meaning of the program language), menus, command code, and response time.

Human Factors Engineering (or Ergonomics as it is often called) is the profession dedicated to the study of the interactions of people with the hardware and software of equipment and the impact of the environment upon this interface.

When the friendliness of such an interaction is evaluated, its usability is usually quantified by monitoring performance (usually in terms of response time and/or error rates) of equipment operators (users). Other variables which also are evaluated are search time, identification, legibility, readability, and information processing time. Most of these items are relatively easy to measure and quantify in an objective manner. Other assessments are more difficult to quantify and are usually measured subjectively (e.g., using rating scales).

The purpose of this paper is to report recent research by Human Factors Engineers in areas that are of major impact to users of computer terminals. These areas of interest are: terminals (i.e.,displays, screen treatments, keyboards), furniture (i.e., chairs), environmental considerations (i.e., lighting), and software (i.e., information coding with emphasis on the use of color).

Furniture

Recent research has demonstrated that correctly designed furniture may have far more impact on overall comfort and stress symptoms than previously believed.

Until recently, most chairs for workers consisted of body support in the form of a seat pan with a back support (usually restricted to support the middle of the back). The better designs allowed for adjustment of the height of the chair seat, and possibly, the angle of the back. Some even provided an adjustment of the height of the back.

Most of the mechanisms used to control these adjustments have historically consisted of manually operated knobs or cranks. Most of these controls have been "less than friendly" due to excessive torque resistance, incompatible in design with the shape of the human hand, and, even at times, unsafe (e.g., pinch points or knuckle crackers).

The last decade (particularly in Europe) has seen a move toward "the humanization" of furniture, especially work furniture. Major design moves in this activity have occurred in the last five years. Some of these changes have directly resulted from the studies conducted by Human Factors Engineers (Ergonomists). These studies (both laboratory and field) have demonstrated a number of interesting factors regarding the use of chair adjustment features. The first is that many users do not realize the existence, relation to overall comfort, and/or the range of variability of the adjustment mechanisms of their chairs. The second is that although they may be aware of these items, they may not use them due to some of the "less than friendly" characteristics listed above. The third is that they may not be correctly adjusting the chair to the most advantageous position for their body dimensions and task requirements. The forth is that even if the subjects correctly adjust the chair, they often complain of postural stress because most chairs do not supply properly distributed support. The major conclusion of these studies demonstrate the need for adequate education in the proper adjustment of chairs.

These studies have shown that a properly designed chair should provide a seat and back shaped to the body and that the back support should extend the whole length of the users back.

Current recommendations are based on the findings of these recent studies. The height and angle of both the seat and back should be adjustable. The height of the seat should be adjustable both in an upward and downward mode. The seat should also have a tiltability feature with the pivot at the back of the seat, providing the ability to raise or lower the front edge of the seat from a position horizontal to the floor.

Field studies have also indicated that the normal sitting posture of workers is not that as shown in textbooks or brochures on furniture (particularly work chairs). These studies have demonstrated that people adapt and change their sitting posture to one that is the most comfortable to them. This is usually not the straight upright back perpendicular to the thighs which are horizontal to the floor and perpendicular to the lower legs. In contrast, for instance, studies of VDU workers show that they lean back in their chairs resulting in a back/thigh angle greater than the 90 degrees seen in references and extend their legs in a more forward position. Researchers have explained that this position more closely approximates a standing position which induces less strain than a sitting position on the lower lumbar region of the back.

The results of these studies have even more extensive conclusions. Because of this backward lean of workers, the height for typing on a keyboard has been demonstrated to be higher than currently believed correct. For example, Germany has legislated that the center row of the alpha section of a keyboard should be at 30 mm (1.14 inches) from a table top which is located 720 mm from the floor. This dimension was based on postures as shown in standard text books. Field and laboratory studies have indicated that users not only prefer to key at a height higher than these dimensions but that they perform better and incur less muscle stress.

There are other results that also should impact the design of VDT equipment. Studies have demonstrated that although a palm rest located along the front plane of a keyboard may provide a resting support for the wrists, it

impedes the performance of a high speed typist. This appears to be due to the interference of the downward movement of the palms of operators. Therefore the need and preference of a palm rest is dependent on the task or application.

Another factor in the "friendliness of furniture" is the location of equipment, particularly the display terminal on a desk top. Most displays are limited in their location by the space available on the desk surface and the dimensions required by the technology. In most displays, this space is occupied by the length of the cathode ray tube inside the VDT cover. These restrictions usually result in the display being located so that the screen is 50 cm (20 in.) from the user. Field studies have shown that when given increased table top depth, users prefer to place the screen between a range of 61 cm (24 in.) to 93 cm (37 in.) from them. The average preferred distance was 76 cm (30 in.), which is 10 inches greater than the distance at which most displays are usually located.

This finding has interesting implications for visual comfort. Most reading glasses are prescribed for a viewing distance of 30 cm (12 inches). Some of the reasons for reported complaints of visual discomfort at a VDU are based on the fact that this prescription is incorrect for typical screen viewing distance (typically at 50 cm). If users are given the opportunity to move VDU's further from them (with increased table top space or a change to a new technology like a flat panel display), this problem will be confounded because the commonly prescribed focal length for lenses will be even more inappropriate.

The ability to tilt or swivel a VDU also enhances the "friendliness" of the system. Most of the advantages of these adjustments are due to the reduction of reflections or glare spots from the screen. Others are based upon providing a comfortable viewing angle for operators of different heights.

Some reflections can be reduced with the use of screen treatments (like etched, micro mesh, or coated filters). Others can be significantly reduced with the proper design and use of office lighting. The following section will address these factors.

Screen Filters

Many of the widespread complaints of VDT operators relating to vision are the reflections visible on the screen surface. Reflections are images that are mirrored from the surface of the screen. Glare spots are bright areas that are a brighter illumination than the ambient screen brightness. Most glare spots are a result of illumination sources in the environment, like windows or flourescent or task lights that can be seen reflecting off the screen.

Reflections and glare spots are a source of visual distraction and reduce the contrast of the characters on the screen. This phenomena is particularly relevant to screens with negative video (light characters on a dark background). One of the reasons preferences have been demonstrated for positive video (dark characters on a light screen) is that illumination reflections are not nearly so visible because they blend into the white background and do not reduce character contrast. Also, the transfer of familiarity with printing on paper may have more influence on user's preferences than any other factor. However, the light background may add more glare to the user's visual environment than a dark screen. In fact, the perception of flicker on positive video is greater than negative video. In addition, characters on positive video look smaller (thinner) and usually cannot be recognized at as far a viewing distance as those on negative video. This occurs because of the "blooming" effect of white or colors highly desaturated compared to dark colors.

There are currently available several methods of reducing reflections and/or glare from screen surfaces. These include: etching or coating the surface of the screen, or placing a filter such as nylon mesh, tinted plastic, circular polarizer, or combinations placed onto or over the screen. The most popular devices appear to be the etched, coated or mesh filters. There are advantages and disadvantages which should be considered prior to deciding which one to use for a particular application.

An "etched screen" is one that has been textured to reduce reflections and glare. The treatment is effective as a diffuser of reflected images but the inherent result is also a diffusion of the characters on the screen. Some etched screens are also color tinted which results in the screen looking darker and may appear to enhance the

character to screen contrast.

The "mesh" technique is the placement of a fine nylon mesh over the face of the display screen. The mesh greatly reduces reflections and glare, but there are four disadvantages: collection of dust between the fibers of the mesh, fragility, limited viewing angle, and geometric wave patterns (bars or rings).

The "coated" filter is one in which an optical treatment is applied to a substrate placed over the screen. This method is also very effective in reducing reflections and glare. The major disadvantage is the susceptibility of the filter to fingerprints which are visual distractors and reduce character to screen contrast. Fingerprints must be removed from the screen with a cleaner.

There has been some research conducted to evaluate the impact on viewability of these different screen treatments. Studies have demonstrated that the coated filter results in the least character distortion. Research currently being conducted should indicate which filters should provide maximum resolution with particular display technologies in different illumination conditions.

Utilization of Information Coding Methods

Research and experience has shown that when information is presented in a coded form it can enhance transfer between the display and the user. The coding techniques commonly utilized to accomplish this enhancement are: blinking, underscore, brightness, reverse video (polarity), alphanumerics, shape, clustering, graphics and color. A vast amount of research has been conducted on the advantages of these different types of coding techniques. Selection of the proper coding has been demonstrated to depend on: the application (task), the display technology, the unique characteristics of the code, the ambient illumination conditions and the user's visual characteristics, education and experience.

The superiority of a particular coding technique is primarily dependent upon the "task" to be performed. This includes such imbedded considerations as: relevancy, redundancy, consistency, rationality, translation, and association.

The "relevancy" of a code conveys its appropriateness for a particular application. For example, characters coded in red are more appropriate to display a dangerous condition status (e.g., error message, overdrawn customer, or radiation leak in a nuclear power plant). Less obvious relevant types of coding include the use of "clustering" and graphics to show weather conditions on a map or "brightness" to segregate recently input data from previously stored data.

"Redundancy" is the duplication of coding to increase the conveyance of a meaning. Incorporating an additional coding technique further enhances the focus of attention attracted by the change in the image. Therefore, if a blinking image is further coded with a color different than the other images on the screen, the attention power of the change would be even more enhanced. This occurs because research has demonstrated that attention is most quickly attracted by sequentially fluctuating the illumination of an image, rather than by increasing the illumination to a steady state.

Whatever coding technique is selected, "consistency" has been shown to be one of the most important considerations of system usability. If a code is inconsistently applied, the result will be a decrease in user performance, increase in mental load, frustration, training and retraining time. If reverse video is used for input fields, for one application, underscore should not be used for input fields in another application in the same user's environment.

The coding technique selected should reflect the "rationality" of the user's expectations and/or understanding of the code. Much of this consideration is based on the educational and cultural familiarization of the user. What may seem very rational to a computer programmer (e.g., machine language) may not make any sense to a user unfamiliar with computer technology. For example, the command code "execute" may appear very irrational to a naive user. Recent advances in friendly software has enhanced rationality of some languages.

The "association" value of a code can also determine its impact on overall usability. As with the use of rational codes, this consideration is largely culturally or educationally determined. Certain colors have high

association values, (e.g. red for danger or stop, green for safe or go). Also, spatial groupings usually mean that similar kinds of information or items belonging to a common classification are located in a similar space or plane. Certain types of graphics are more associated with displaying scientific information; others may be more associated with displaying business status.

The "display technology" can also influence the usability enhancement of a coding methodology. The technical considerations include chromatic characteristics, saturation, brightness and contrast. Some displays are limited to the number of coding techniques they can produce. Others are limited by the quality of the images. Some displays do not provide controls to properly adjust brightness or contrast. Only recently have displays been sold that produce high-quality positive video.

Some of the "characteristics of the user" considerations have been previously mentioned. These include: training, associations, expectations, and cultural differences; others include perception, visual deficiencies, transfer, and cognition. One consideration often ignored in use of displays is the age of the user. Elderly workers are less sensitive to saturated colors on the low end of the electromagnetic spectrum. Therefore, saturated blues should be avoided to encode important information viewed by this user group. A sizable percent, (8% or more) of the male population is color deficient. Care should be exercised that colors susceptible to these deficiencies be avoided or desaturated, and in particular, colors easily confused should not be grouped when used to differentiate information. The ability to detect detail is dependent on a particular visual anomaly. Individuals with cataracts are reported to perceive information more easily in negative video (light characters on a dark background), whereas individuals with glaucoma may more easily see images displayed in positive video (dark characters on a light background).

The "ambient illumination" can impact the effectiveness of the coding technique. For example, information on color displays is usually more clearly perceived in ambient illumination (200-300 lux) lower than the average office (500 lux), because bright lights make the colors become "more washed out" (the contrast appears reduced). If the ambient illumination is excessively high (e.g.,

1200 lux), the contrast, and hence the legibility, may even be severely degraded on a monochromatic display. In addition, the amount of reflections from the screen surface should be evaluated. If there are many reflections on the screen from overhead lighting or light colored objects (like white shirts), the utilization of illumination coding techniques like brightness, underscore and blinking may not be detected as quickly, if at all, as other codes like clustering and shape. Again, this is due to a reduction in the contrast of the characters to the screen background.

Color Coding

Most of the research comparing color coding techniques has been historically conducted using media other than visual display units. For this reason, many of the findings may not be applicable to VDU images because of the inherent differences in character generation and therefore their impact on visual processing. Displayed images are refreshed and, in general, have a much poorer image resolution (e.g. blurry edges) than a printed image. In addition, environmental images are not reflected from the face of a document like they are from a screen, although glare may be emitted from a document (especially with high gloss paper) as well as reflected off a screen. The findings reported in this paper are mostly a summary of historical research in color coding and, therefore, may not be valid when considering the use of color for display applications. Further research is necessary to evaluate the relevancy of past research and the similarity of findings from studies of color on documents to color on displays.

General research in color coding has demonstrated that color: a) is superior in identification when compared to shape, size, and brightness; b) is easier to distinguish in the periphery of the visual field (off center viewing) than either size or shape coding; c) is learned faster and remembered longer than information presented achromatically; and d) requires fewer eye movements to locate information. In regard to the last finding, color may be viewed as a technique to alleviate visual discomfort because a reduction in excessive eye movements relieves muscle fatigue.

When evaluating the cognitive aspects of color, it has been shown that the brain processes information more effectively with color coded information than when information is coded alphanumerically. Also, response time was reported to be faster when viewing information coded in color compared to shape coding or achromatic information.

There have been some recent studies evaluating the enhancement of usability when using a "multi-color" display compared to a single color (monochromatic) display. Most of the research has been conducted for military applications, but recently, more scientific analysis has been conducted on utilization for office and/or manufacturing applications.

Studies conducted comparing these two types of displays demonstrated that the multi-colored display reduced training time, improved readability, reduced search time, increased performance, and is more preferred. It has been demonstrated that the advantages of a multi-color display are most obvious when information is presented in a complex format. The types of information that can best be color coded vary between applications. Based on the research conducted thus far, the information that best lends itself to improved processing by color coding is that which should be remembered, located, identified, changed or corrected. In addition, fields of information (e.g. input or output) are also good candidates for color coding.

In general, the following colors have been shown to be most successfully used for the indicated types of applications: red - danger, warning, errors; blue - borders; green - normal data entry; white - output information; yellow - control areas. Certain color combinations are more easily seen than others, and should be utilized when information is to be segregated. These include: red/green, red/blue, blue/yellow, or combinations like red/blue/yellow. Colors that are similar in hue but different in saturation can be used to display a subtle difference. Examples of these are: blue/turquoise, red/pink, green/chartreuse, yellow/white.

One of the most important considerations to remember about any coding technique is that it can decrease the usability, and degrade performance if inappropriately used. When problems occur, they can generally be traced to the

following reasons: inappropriate application, improper format, lack of user training, inconsistency, confusion and/or perceptual problems. The solution to most of these may be obvious. However, perceptual problems such as: identification, discrimination/flicker, glare, confusion, adaptation, and varying distance effect may not be as apparent. Many of these problems can be easily remedied. For instance, the perception of flicker can be reduced if the brightness of the characters is reduced. The varying distance effect (e.g., blue images look closer than red ones) can be alleviated if the ambient illumination is increased.

General guidelines for the use of color in multi-color displays recommend that the utilization of color be consistent, the choice be conservative, and that colors are best grouped into large areas as opposed to small spot identities.

In summary, the advantages of a colored display have been shown to aid in: attracting attention, locating, identifying, prioritizing, defining, creating realism, improving associations, segregating, grouping, chunking, memory, discrimination, simplification and aesthetics. These advantages have been demonstrated to result in: reduction of eye movements, a higher level processing, improved perception, performance, and usability.

Conclusion

User friendly systems are the result of good and appropriate research and the incorporation of these findings into product design. However, the friendliness is also a result of the proper use of the product (or system) and thus, to a large extent, depends on the education and motivation of its users.

This paper was presented to show recent research that has demonstrated how terminal design factors can be utilized or modified to enhance user friendly VDT systems.

References

1. Banks, W. W.; Gertman, D. D.; Petersen, R. J.; "Human Engineering Design Considerations for Cathode Ray Tube-Generated Displays," NUREG/CR-2496, EGG-2161, U.S. Dept. of Energy, EG&G, Idaho, April 1982.

2. Emmons, W. H., and Hirsch, R. J.; "Thirty Millimeter Keyboards: How Good are They?" Proceedings of the Human Factors Society, Seattle, Washington, October 25-29, 1982, p. 425-429.

3. Grandjean, E.; Hunting, W.; Pederman, M.; "A Field Study of an Adjustable VDT Workstation and their Effects on Body Postures and Subjective Feelings." Paper presented at Human Factors Society Conference, Seattle, Washington, October 25-29, 1982.

4. Grandjean, E., and Vigliani, F.; Ergonomic Aspects of Visual Display Terminals. Taylor and Francis, Ltd., London, 1980.

5. Helander, M. G.; Billingsley, P.A.; Schurich, T. M.; "A Critical Review of Human Factors Research of Visual Display Terminals," Technical Report No. CRG-TR-82-004, Bell Labs, New Jersey, July 1982.

6. Jacobsen, K. D., "Optical Measurements of Display Products," SID Digest, 1982, P274-275.

7. Kolers, P. A.; Wrolstad, M. E.; Bourma, H.; Processing of Visible Language, 1 and 2, Plenum Press, New York, 1980.

8. Mandal, A. C., "The Seated Man," Applied Ergonomics, 1981, 12:1, p. 19-26.

9. Robertson, P. J., "A Guide to Using Color on Alphanumeric Displays," IBM Technical Report, Hursley, England, June 1980.

10. Robertson, P. J., "Review of Colour Display Benefits," IBM Report #HF056, Hursley, England, January 1982.

THE USER'S ROLE IN SOFTWARE DESIGN

Stan Swete, Software Engineer
ASK Computer Systems, Incorporated

INTRODUCTION

The past decade witnessed important changes in the type of programming performed and the ways in which this programming was managed. The emphases shifted from systems type programming (compilers,operating systems,etc) to applications type programming. The management of the programming effort changed from an ill-defined art to a structured approach more closely resembling a science.

While much has been accomplished in development of design methodologies, one important problem unique to applications type programming continues to be overlooked or improperly addressed. The problem is the method developers use to determine the needs of the applications' end user. Industry journals recognize the symptoms but do not correctly identify the source of the problem.

The problem concerns information gathering, not information management. The solution must involve the user's role in design. Current efforts seek to improve the way user input is incorporated into the product but not the way user input is originally solicited.

This paper will demonstrate that this problem exists and will explain why it is overlooked today. Finally, it will propose ways in which the user's role in design can be amended to solve the problem.

A BRIEF HISTORY

In a keynote address given at the IEEE Spring '76 COMPCON, Robert McClure offered a brief synopsis of the programming effort since 1956 [8]. In the first decade the major contributions were generalized programming tools (high level languages like FORTRAN and COBOL were among the most important of these tools) and hardware developments which allowed programmers to begin working on-line.

The second decade's contributions were methodologies instead of products. According to McClure, in this decade, "... software stopped being fun and games and was recognized as a serious business."([8], page 7). These methodologies were a response to the problems encountered in early attempts to design large software systems. This decade may have signalled an end to the fun but the interest in programming as an industry continued to swell.

Four events from this second decade marked the beginning of the structured programming movement. These events continue to influence thinking and writing about software development today. The four events (in no particular order) are:

1. The NATO conferences of '68 and '70 which introduced common problems with design of systems and coined the term "software engineer".

2. Publishing of The Mythical Man Month by Fred Brooks in 1975 [3.]. The book related actual problems encountered in a large design effort and exposed shortcomings of the (then) current theories concerning project management.

3. Publishing of The Psychology of Computer Programming by G.M. Weinberg in 1971 [12]. This book attempted to describe the demands the programming task makes upon programmers and the characteristics programmers should have to best meet these demands.

4. Publishing of the article, "Chief Programmer Team Management of Production Programming" by F.T. Baker in 1972 [2]. This article introduced an effective organizational structure for programming projects.

This second decade also saw the programming effort shift from systems programming to applications programming. This important occurrence introduced the non-professional end user into the development picture.

McClure's history of software development remains fairly complete today even though it is seven years old. Much has been accomplished of course but the list of major changes still stands at two:

1. First decade advances brought the programmer closer to the machine.

2. Second decade advances brought the untrained user closer to the machine.

CURRENT DESIGN METHODOLGY - AN EXAMPLE

The results of second decade research were new methodologies outlining better ways to create software and to manage the creation of software. The overall result is often called structured design. A structured approach to design breaks up the design task into standard sub-tasks. Each sub-task should have well defined starting and ending points and specific required results. The better software developers today realize significant improvements in product reliability, product maintainability and general project accountability by employing some type of structured design method [6], [10].

A typical application design method has an initialization stage during which the proposed project is roughly defined, the project's market is assessed, and some type of go/no-go decision is made. This stage concludes with a rough schedule of the rest of the project's development stages.

The next stage is an attempt to determine general project requirements and user needs. This stage typically consists of meetings between users and developers. The goal of these meetings is to generate a list of requirements (requirements definition document). From this document, a conceptual design is developed. In the best cases the developer walks through the requirements definition and the conceptual design with the user. This stage concludes with a list of project requirements which has been approved by the user and a refined schedule of remaining work.

After the project definition stage, the project team turns its attention to specifying the input and output requirements of the system. This stage is often called external design. It results in a set of I/O specifications, a functional design of the system and another revised schedule of remaining work.

The next stage is internal design. At this point the project team begins to think of the system's functions as groups of programs. The database (and/or other storage facility) is designed and a specific schedule for programming, debugging, testing and release of the system is developed. This stage results in a finalized schedule of remaining project activities and a list of programs comprising the project.

We will concentrate primarily on the requirements definition phase of design methods. This sample methodology should provide general information and define some terms to be used later.

THE PROBLEM WITH DEFINING USER REQUIREMENTS

There are difficulties with performing each of the phases in the design methodology mentioned earlier. Creating applications systems is a challenging task. However, while other problem areas stimulate research which generates more acceptable methods, the problem of determining user requirements achieves recognition but not results.

Two recent surveys explored current software design problems and the solutions commonly proposed for these problems. In both cases, the problem of determining user needs was widely recognized as a serious shortcoming of current design methodologies. In both cases the solutions proposed were either too general to be of use or did not properly address the problem.

John Lehman [7] gathered his data from fifty-seven actual projects. Lehman's survey centered on managerial and programming practices used, problems encountered and results obtained. One area of questioning concerned preparation of requirements specifications. The comments by survey participants indicated that this area was an important concern to most managers. Managers' comments, "... reflected the complaint that specifications are all too frequently incomplete, ambiguous, inconsistant and plagued with seemingly arbitrary changes,"([7],page 129). As for solutions to the problem, "... virtually every comment stressed the need to establish a closer relationship between the system user and functional analyst,"([7],page 129).

Thayer,Pister and Wood [11] conducted a different type of survey. Their's concentrated on identifying which (of a list of twenty) problems were considered important and on determining the solutions most frequently proposed for those problems. One of the twenty problem areas concerned requirements specifications problems. The survey was given to several hundred people from various professions. An extremely high percentage (over 90%) of every profession rated the problem of requirements specification as being important.

The survey then went on to determine the most commonly proposed solutions for the problem. There was no single solution commonly put forth for the problem of requirements definition (most entries fell into the "other" category). The second most commonly proposed group of solutions advocated the use or enforcement of existing techniques. The third most popular solution category, comprising only 2.8 % of total responses advocated increased communication between user and developer ([11],Table VIII page 340).

WHY SUGGESTIONS ARE LACKING

Research efforts recognize the problem of effectively documenting user needs but do not identify solutions which directly address the problem. In applications work, the requirements definition phase is crucial. This initial phase must be performed well if the project is to have any chance of succeeding. The only sources of information available are the user's knowledge and the developer's knowledge. In my opinion, the lack of effective progress in the area of user requirements definition results from a failure to recognize the importance of this phase of design and a reluctance to consider new ways of soliciting the user's knowledge.

The tendency to overlook the importance of requirements definition to the product probably has some historical basis. As mentioned earlier, the emphasis on applications programming is a relatively modern phenomenon. Early programming efforts had to concentrate on getting the package to run because just getting it to run at all was difficult. New programming tools have changed this situation. It is now less difficult to get a piece of code to run. However, the old values still remain. The primary goal is still to make the product work. After you get it running you can add in user friendliness, comments and other options to make the package more attractive.

This type of thinking is completely inappropriate for applications programming. The application system has to be designed to fit its intended user. This fit cannot be added in as an afterthought. It must be a basis for design. It must be defined before design can begin. Failure to define needs first insures imperfections in the final product. The realization that these imperfections cannot be ironed out in later design phases should lead applications designers to place more emphasis on early definition of needs.

The reluctance to consider new possibilities for the user's role is the main reason that the problem of requirements definition is frequently addressed improperly. Researchers who attempt to find solutions to the problem concentrate on improving methods of expressing the list of requirements (requirements definition languages) and on improving methods of implementing the list (programming techniques) but not on improving methods to generate the list. The user is supposed to present the problem to the developer and is then supposed to go away and come back when the system is finished. Unwillingness to alter this situation is largely the result of a conceptual block on the part of the developer.

Adams [1] defines conceptual blocks as, "... mental walls which block the problem-solver from correctly perceiving a problem or conceiving its solution,"([1],page 11). In the case of applications software development, the developer is blocked by a view that the user's role must be like that of any other customer just as the developer's role is like that of any other producer. The user pays for solutions, not for the chance to work on problems.

This common view may apply for some industries but it cannot apply to applications development. Since the user's input determines success or failure in the resulting sytem, the applications user must be given a different role. The first step to defining useful roles for the user is to eliminate the unnecessary restrictions which current conceptual blocks impose on the user's role.

The failure of current research to properly attack problems concerning the user's role is indicative of a need for new thinking about applications-specific problems. Applications programming is a new field with unique problems. Research efforts today still focus on problems and approaches proposed in the early '70s by Brooks [3], Baker[2] and others. Such efforts are not wasted but they do not address the problems which applications programming does not share with systems programming. This is not surprising since the original thinking behind these efforts was in response to systems programming problems.

REDEFINING THE USER'S ROLE

In order to improve requirements definition the developer must commit to allowing the user multiple chances to refine the requirements list. The current method of developing requirements' definitions consists of an effort by user and developer to create an initial list, an intermediate period to allow the developer to finalize the list and a final meeting to go over the final list with the user.

This method does not succeed because it places an impossible demand on both the user and the developer. It expects the user to convey a list of all the important details of the application in one meeting. The method then expects the developer to fill in every gap in the user's list. The only feedback user and developer receive is the finalized specifications statement. The user will not be able to competely define the application at the initial meeting. The developer cannot know the user's specific application to the extent the user does. Neither party will be reminded of all items left off the list by looking at the list again. The method is bound to generate incomplete results even with multiple iterations because of

the lack of effective feedback. Methods which allow the user and developer to iterate based on effective feedback will provide more complete requirements lists.

Because users are not programmers, it is impossible for them to determine the programmatic results of requirements information. Until they see some of the results of their suggestions they cannot know whether they have been too specific or too general or even whether something important has been left out. Martin [9] suggests prototyping of applications systems as a way to provide feedback to the user in the early going. The user and developer meet to draw up an initial list of requirements. The developer then uses a high level query language to develop a prototype (or mock-up) of the proposed system. The prototype serves as, "a focus for debate which helps to ensure that they (user and developer) are both talking about the same thing,"([9],page 64).

After working (interactively) with the prototype the user makes new suggestions based on a new state of knowledge. The developer incorporates these suggestions into the prototype and lets the user experiment further. These steps are repeated until user and developer are satisfied that all necessary requirements have been included. At this point the prototype is the requirements document to be used for further development ([9],page 64).

The idea of using prototypes is one way of insuring a more effective initial phase of design. Other methods can be used (using walkthroughs of external design work as a source of feedback for requirements' definitions is one example). These methods are effective because the user works with partial design results to support his/her own knowledge of the application. The important point behind such methods is that the requirements definition phase and external specification phase of design must be performed simultaneously instead of sequentially. The design effort will enter the internal design phase with the required planning documents (a requirements list and a functional design) but these documents will represent a more realistic view of user needs.

CONCLUSION

From the viewpoint of the user, applications programming can be described historically as a field which started as a mysterious art, is currently a valuable service and shows promise of becoming a personal tool. There often seems to be an awkward period in the life of any evolving process. I think it is fair to say that applications programming is currently in its teens. User and developer communicate well enough to generate

excitement about the proposed system's potential. However, the communication is often incomplete enough to allow for surprising and disappointing results.

Current trends indicate that this state is not permanent. Given cheaper, more powerful machines and higher and higher level languages, users will one day design packages for themselves thereby eliminating the frustration of communicating needs to another party [9]. Since this day is not here yet, software developers may make the first move toward applications programming maturity.

This paper has mentioned changes in the current design process which can help the developer determine users' needs more effectively. The important idea behind any change in the design process is that the developer must work with the user in order to best work for the user. Well targeted applications systems must start with this commitment. For both the user and the developer, it is a task which may not be technical but which is certainly not trivial.

REFERENCES

[1] James L. Adams, Conceptual Blockbusting. Stanford, Ca., The Portable Stanford, 1974.

[2] F.T. Baker, "Chief Programmer Team Management of Production Programming", IBM Systems Journal, vol.11, no.1, 1972, pp. 56-73.

[3] F.P. Brooks, The Mythical Man-Month. Reading, Mass., Addison- Wesley Publishing Company, 1975.

[4] Douglas Comer, "Principles of Program Design Induced from Experience with Small Public Programs", IEEE Transactions on Software Engineering, vol.SE-7, no.2, March 1981, pp.169-174.

[5] M.A. Goetz, "Advanced Commercial Applications in the 80's", Datamation, 1979, pp. 104-108.

[6] David P. Laughlin, "Software Engineering Standards And Tools: The System To Produce Reliable Software", IEEE 1981 Software Engineering Standards Applications Workshop, Aug.1981, pp. 32-38.

[7] J.H. Lehman, "How Software Projects Are Really Managed", Datamation, Jan. 1979, pp. 118-129.

[8] Robert McClure, "Software--The Next Five Years", Interface Workshop on Software Engineering, New York, Springer-Verlag, 1976, pp. 5-9.

[9] James Martin, Applications Development Without Programmers. Englewood Cliffs, New Jersey, Prentice Hall, 1982.

[10] P.W. Metzger, Managing a Programming Project. Englewood Cliffs, New Jersey, Prentice Hall, 1973.

[11] Richard H. Thayer (et al), "Major Issues in Software Engineering Project Management", IEEE Transactions on Software Engineering, vol.SE-7, no.4, July 1981, pp. 333-342.

[12] G.M. Weinberg, The Psychology of Computer Programming. New York, Van Nostrand Reinhold, 1971.

ENTITY RELATIONSHIP ANALYSIS

A METHODICAL METHOD FOR IMPLEMENTING RELATIONAL DATA BASE ON THE HP 3000 WITH IMAGE

RAY THOMAS
TEXAS MUNICIPAL POWER AGENCY

PREFACE

Prior to delving into implementing "relational data base" with IMAGE some discussion is required to provide some background information and to establish the boundries of the discussion. This section attempts to establish those boundries. First and most important is the definition of "relational data base." Relational data base is defined to be a specific construct of logical data definition, not a relational data base management system. Relational data base refers specifically to the logical manner in which the data is defined to the data base management system (DBMS). This definition and how to achieve it is the purpose of this paper. The relational data structure which is obtained in the "logical data model" portion of this paper can be implemented on any DBMS; the physical model is tailored to the IMAGE environment.

The characteristics of relational data structure were defined by Dr. E.F. Codd in his paper on relational data base.[1] The process which he describes for reaching relational structures is called "normalization". To achieve "relational" data structures, data has to be at least in "third normal form." "Third Normal Form" data bases generally have no redundant data except for keys and have no calculated data maintained. Data in relational structures may be defined to even higher levels of normalization (i.e., fifth or sixth normal form). Each level of normalization solves data structure problems which occured in the previous level of normalization. Normalization is generally achieved through a reduction process of all known data requirements.

My analysis of Third Normal Form data structures provided the following findings:

1. The data characteristics of third normal form, because of their origin in relational calculus, are too difficult for the average business systems designer or programmer to understand.

2. Third Normal form data structures for large data bases are "source level" data. That is, when the data is completely normalized there are no calculated data items maintained.

3. The totality of all tables and what data is in them is difficult for users of large data bases to comprehend.

4. From a business point of view, higher level data structures are apparent in large third normal form data bases. This structure is the "Entity" structure. Business Entities are identifiable as groups of tables and those tables have specific characteristics. I found that true logical "relationships" exist only between business entities and that the tables could be configured into hierachical data structures which described entities. These concepts have proven to hold true in all data requirement cases.

5. For real-time access requirements to a relational data base, direct (keyed) access to the data base coincides with entities and the hierachical structure of data within the entities.

On the physical data management level, Dr. Codd's definition of relational structure is correct and essential to the creation of a DBMS which can accomodate the data handling requirements of relational data structures. However, business data problems require a different method for defining data relationally.

The level of normalization achieved through the Entity Relationship Analysis approach is different from those currently defined by Dr. Codd and his associates. For the purposes of this paper, the level of normalization will referred to as "Extended Third Normal Form." Extended third normal form has all the same characteristics of third normal form plus some additional features. Since the third normal form structure is derived from Entity Relationship Analysis, there are some differences in the way certain terms are used. The most notable differences are as follows:

1. In Third normal form, data is maintained in tables. By Codd's definition, each table is an entity. Entity in the Extended Third Normal form is a "thing" about which data is kept and may be described by more than one table.

2. In the Extended Third Normal Form, relationships exist only between entities. Data structures within entities are hierachical.

3. The method of deriving the normalized data base is based upon analizing data about business entities, rather than normalizing the data required to support known output requirements.

4. The relational tables are "indexed" by entity providing a structured means for users to comprehend large relational data bases.

There are a number of other differences: however, these are the major conceptal ones.

In order to support relational data structures, certain functional capabilities are required. In the HP 3000 and IMAGE environment, a number of software tools are necessary to support the relational processing. The primary tools used to implement the relational data structure on an HP 3000 in a prototype environment were:

1. IMAGE - IMAGE is used as the underlying file management facility. This facility is transparent to the application progammer just as indexing facilities generally are in pure relational DBMS'S. IMAGE recovery facilities are used to protect data integrity.

2. Quick, Quiz, and QTP - Quasar's "Q" languagers are used as the high level language to support the relational DBMS.

3. QSCHEMAC - Quasar's compiled Qschemac which drives the "Q" languages is used as the relational DBMS. All program code interacts with this facility rather than IMAGE directly. This facility is also used as the prime security control for the relational data base.

4. Integrated Data Dictionary - This is an interactive facility for defining data about data. This facility provides several utilities for managing and manipulating the IMAGE data bases and Quasar's Qschemac. This facility also provides extensive inquiry and reporting facilities which are tailored to Entity Relationship Analysis.

5. ADAGER - This facility is used to dynamically change the IMAGE data bases as required by the relational environment.

Even with this set of tools, all relational functions are not satisfied. However, the system functions relatively well compared to other relational environments.

INTRODUCTION

Entity Relationship Analysis is a structured process for defining Extended Third Normal Form data bases based upon logical understanding of business functional requirements of application systems. This paper assumes that the business analysis has been completed. The process of creating a relational data structure consists of the following three steps:

1. Entity Relationship Definition
2. Logical Data Structure Definition
3. Physical Data Structure Definition

Each of these steps creates a diagram which graphically represents the data structure at that level. Information for these models is loaded into two data models which describe the data. These data models are most conveniently maintained in an interactive dictionary. These models are called the Logical Data Model and the Physical Data Model. The remainder of this paper is presented in terms of the processes required to complete these models. Maintenance to the relational data structure is performed interactively in conjunction with the existing data models defined in the dictionary.

LOGICAL DATA MODEL ANALYSIS

The organization is viewed as a whole, having a single data base. An organization may be defined within the "political" and "technical" limitations as required; optimumly it would be the complete organizational (corporate) entity. This, however, is not always feasible. Application Systems are "logical views" of the complete data base from a particular point of view. It is assumed that the Logical Data Model for the Application system under review is or will be supported by a Extended Third Normal Form data base. The method used to create the single data base is Entity Relationship Analysis. This process creates a logical data base as seen by the "user". Using this methodology a single data base can be created in a building block fashion through the traditional "one application at a time" approach. That is, the organization's total data requirements do not have to be defined prior to building or replacing all the operational systems. Entity Relationship Analysis is used for each application, but the analysis always builds upon or extends the organization's existing logical data model. Logical Data Models are not built for each application. The data gathered in this analysis process is stored in a data dictionary and is available to all development projects through the evalutionary process of application development. The most critical part of this method is the data analysis and the single most important factor in this approach is the point of view for the data analysis; it must occur from the "single data base" concept.

This concept should be understood and accepted by all development projects. Management outside the data processing area will and need not, be concerned as long as "their application" is produced in a timely fashion. (Actually the same principle can be applied to individual applications, but the benefits are not as substantial.)

The work on the logical data model begins after the completion of functional or business process analysis. The data analysis is always top down. To begin the Entity Relationship Analysis, an Entity relationship Diagram is created. Knowledge to begin the analysis is based upon the designers general knowledge of the business processes required to support the application. An Entity Relationship Diagram illustrates all data entities used by the application system and the natural relationships between the entities. Figure 1 is an Entity Relationship Diagram for a Data Processing Project Control System.

Entities are things which the organization maintains data about. They are "nouns". They can exist independently and each occurrence of an entity must have its own unique identifier. This implies that within the complete logical data base, there will be only one occurrence of an entity; i.e. no duplicates. This must be a manageable task from a data maintenance view.

The entities are depicted by single symbol (any symbol such as a square, rectangle or circle could be used). Relationships between entities are expressed by lines and arrowheads. There are only three types of relationships which exist; One to One (represented by a line with a single arrowhead at either end); One to Many (represented by a single arrowhead on the One side and a double arrowhead on the Many side); and Many to Many relationships (represented by a line with double arrowheads on both ends). Entities are things (nouns) about which an organization keeps data. Some entities will be quite clear such as employees, purchase orders, vendors, etc., others will be grayer, but the analysis will make the entities obvious. One saying is, "One man's entity is another man's attribute". The point of view of the complete organization's needs will make these answers clear.

The Entity Relationship Diagram is a single page that has all entities of the system shown. Each entity is tested against the others to identify whether or not the two are related. Each relationship creates two questions: how many entity number two's occur for each entity number one and how many entity number one's occur for each entity number two. This process identifies the type of relationship as One to One, One to Many, and Many to Many. Once all relationships are documented the number of relationships should be minimized.

PROJECT CONTROL SYSTEM
ENTITY RELATIONSHIP DIAGRAM

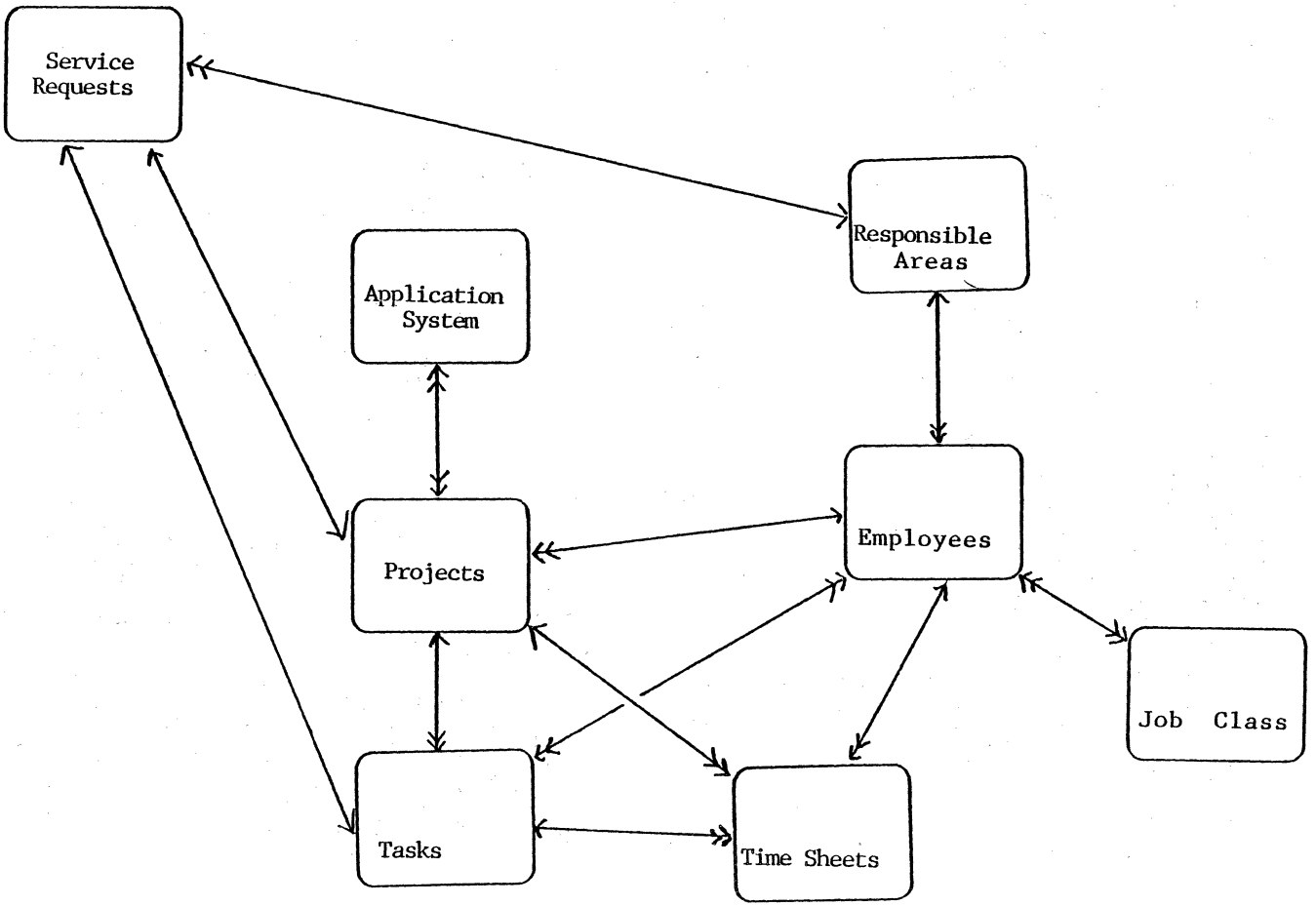


FIGURE 1

Basically, if there are more than two ways to get from one entity to another, the paths should be reviewed to determine if all should be necessary. This determination is not black or white. In some cases, duplication of relationship paths should occur in others they should not.

One special entity is the accounting entity. The Third Normal Form (or source level) for accounting data is a "transaction". This is a logical business "transaction". Each type of transaction is an entity. It has been indicated that no summary data is kept in the corporate data base. For accounting data, this is still true. Some accounting transactions, such as month-end journal entries are summary entries. These are still source level and are recorded like all other transactions. This is an intuitive decision, but it will be clear from the analysis. Another entity which always exists is "day".

Each entity and relationship identified in the Entity Relationship Diagram should be defined in the data dictionary. A paragraph should be written to logically define each entity. The initial Entity Relationship Diagram (ERD) will likely be modified by the additional data analysis of the Logical Data Structure Diagram. Expect to change the ERD; more than likely you will not have defined all the entities you need to support the application. The relationship definitions in the dictionary are more factual in nature and will not require as much narrative or logical description, although the justification or what the relationship represents must be defined. It is this definition process which will clarify many gray areas.

The next step of the data analysis is to define the Logical Data Structure required to support the application. Figure 3 is the logical data structure diagram for the Project Control System. The Data Structure Diagram is initially created from the Entity Relationship Diagram. At this point one must first understand how data (attributes - really adjectives) about an entity are structured. The data (attributes) for an entity always fall into one of five categories:

1. **Non-Repeating Attributes.** These are data elements which occur only once for each occurence of the entity. For example, "date of birth" for an employee. There is only one logical group of non-repeating attributes for each entity.

2. **Repeating Groups of Attributes.** An entity may have any number of groups of repeating attributes. A repeating group is a set of attributes which repeat at the same frequency for the entity. An example of this is line items of a purchase order or addresses for a vendor.

LOGICAL
ENTITY
DATA GROUP
MODEL

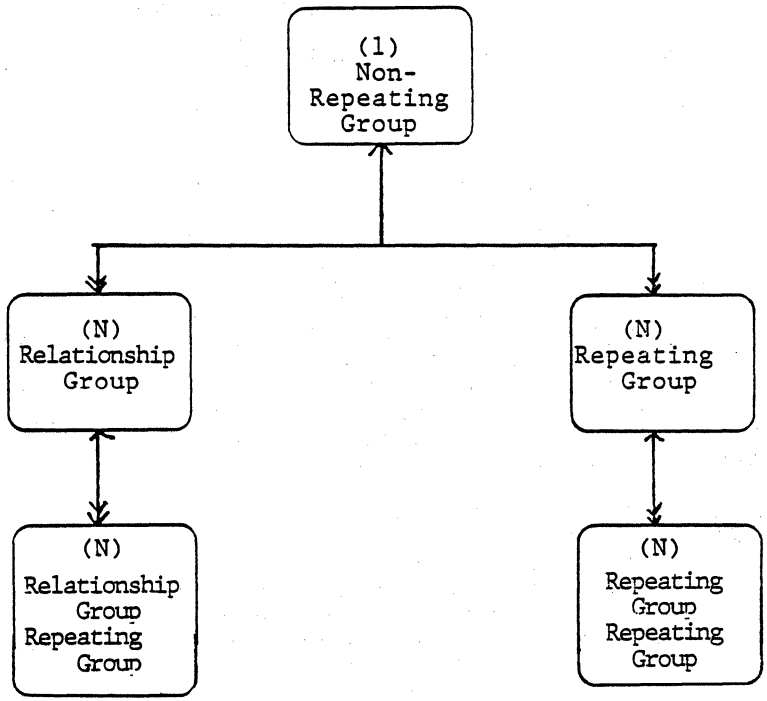


FIGURE 2

3. **Relationship Groups.** A relationship group must occur for each Many to Many relationship an entity may have and/or any relationship which has attributes.

   Each relationship group must contain a minimum of the keys to the entities of the relationship. A relationship may include any number of the entities, but beware of those which include more than two. The relationship group may also contain attributes which modify the relationship of the entities as opposed to either of the two entities. Be sure to make the adjective modify the correct noun, and a relationship is a special noun. It is not an entity, but it is a noun. Relationship groups are "shared" between entities.

4. **Relationship Repeating Groups.** Each relationship may also have repeating sets of attributes. These are to be defined in repeating groups for the relationship. A relationship may have any number of repeating groups.

5. **Repeating Group-Repeating Groups.** A repeating group of data may be modified by any number of repeating groups.

Figure 2 illustrates the conceptual structure of the logical data view of an entity. Under this concept of data definition, the concept of an "occurrence" of an element within a group is eliminated. If an element or set of elements needs to occur more than once, then a repeating group is required. Additionally, none of the groups describing an entity have "summary" data included. All data is maintained as "source" or lowest normal form. Any "calculated data" or "information" is assumed to be obtained through processing. There is no identification of "conditional" data. The conditions for storing data are a function of the program application code which updates the data.

To create the Logical Data Structure Diagram (Figure 3 is a Logical Data Structure Diagram for a Project Control System), start with the Entity Relationship Diagram. Draw one non-repeating group for each entity. Again symbols for data groups are not important, but pick one symbol to represent "groups." Each data group should contain a designation of the type of group Non-Repeating (NR) Repeating Group (RG), Relationship (REL), Relationship Repeating Group (RELRG), or Repeating Group-Repeating Group (RGRG). Next draw in a relationship set for each Many to Many relationship and any relationships which have attributes. Use lines to draw in the One to One and One to Many relationships. The Many to Many relationship groups generate One to Many lines from the entity non-repeating groups to the relationship group. This process

PROJECT CONTROL SYSTEM
Logical Data Structure Diagram

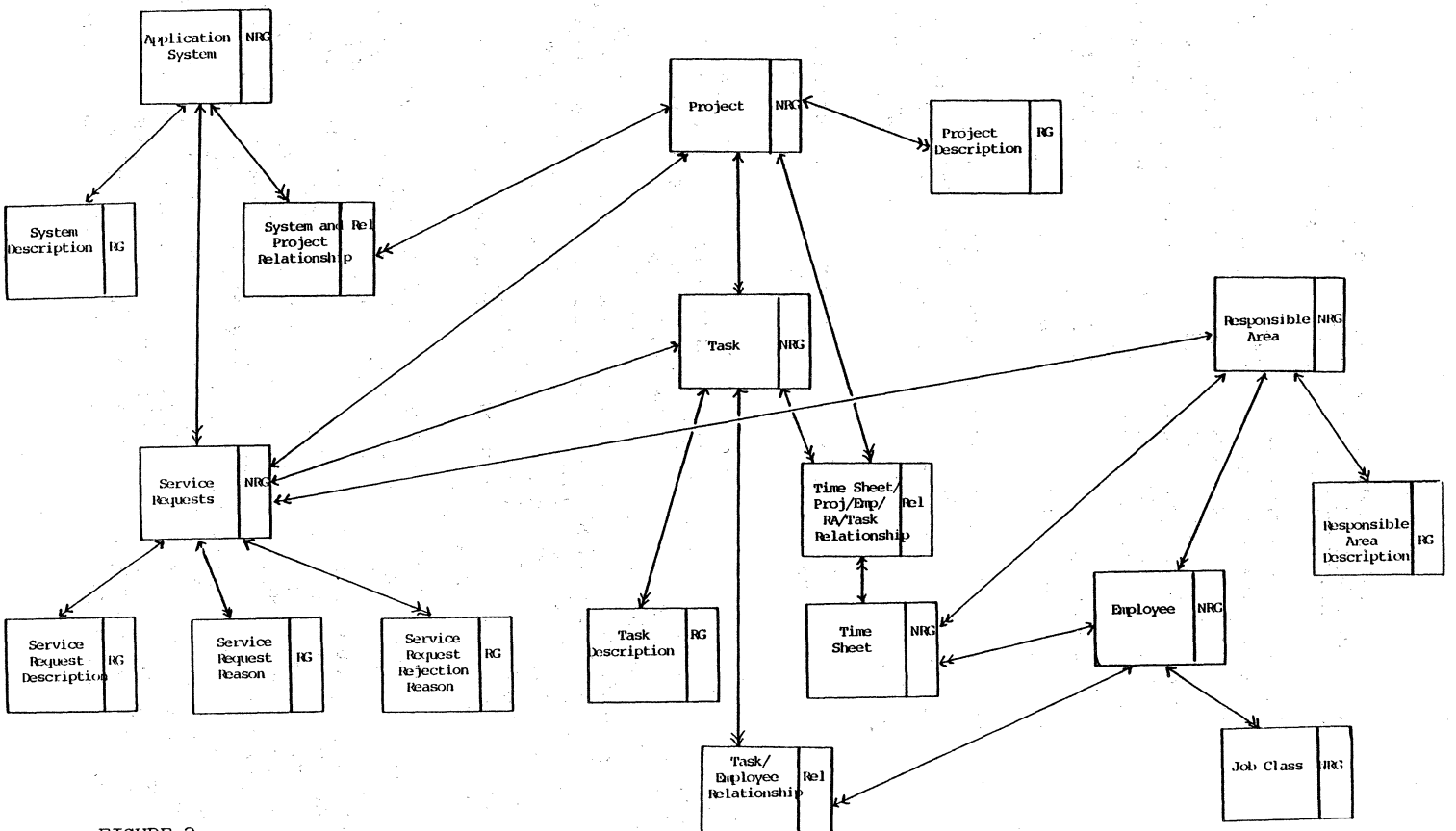


FIGURE 3

is also supported by the data dictionary. In the Entity Relationship Diagramming, all entities were entered into the dictionary. In this step all data groups for an entity are related to it. The relationship groups are defined only once, but related to all entities associated with the relationship. The logical definition of these groups will also clear up gray areas. These definitions are essential to "understanding" the data.

At this point you may begin the detail work of data analysis element/attribute definition and analysis. Begin to associate and define each element with the appropriate group. Start with the "keys". Each occurrence of an entity has to have a unique identifier (unique within the occurrences of the entity). This key must occur in each group associated with the entity. In One to One relationships, the keys of the two entities are maintained in the non-repeating group of the respective entities. In One to Many relationships, the key of the One is maintained in the non-repeating group of the Many. (Assuming the relationships have no attributes). From this point on, begin to identify from the business function analysis what attributes of the entity are required by the functions of the system. Allocate the attributes or elements into the appropriate group types. All data will fit into this structure when normalized correctly. Use the source documents and primary output of the application to identify the elements or attributes required. Do not, attempt to define all the output required for the system to determine the required data elements. Less than 20% of the system's total outputs may be used to determine all the required data elements if this data definition method is used. It is not necessary to have a 100% correct data base design since the physical data structure can be modified using Adager or Dictionary utilities.

Remember in this methodology, you are assumed to be using an automated data dictionary to support your analysis. A record is maintained in the dictionary for each unique entity, data group and element. If the record has already been defined, just relate it to your application system's view of the data model.

Check the dictionary first. In most cases, you will be able to use the description as is; in others, you may have to modify it such that it can serve multiple purposes.

Rules of Thumb:

1. Never build concatenated keys for an entity. Concatenated keys are indications that there are non-repeating attributes within the key which should be in the non-repeating set or that you have defined two entities together. Revise your data structure.

2. Never use a "field" or attribute in a manner such that it has two logical meanings even though one attribute may not occur when the other one does. Give each attribute its own field; space is not that important.

   A classic example of an entity with a concatenated key is the General Ledger Chart of Accounts, Account Number. The components are "non-repeating" attributes. The users will, of course, insist upon concatenated key in this case. In these cases the best approach is to also maintain the components of the key as separate attributes in the record.

The Logical Data Structure Diagram for the application system should be prepared on a single page. Each group should be defined in the dictionary and each should have a logical definition. It will suffice to say, "this is the entity (name of entity) nonrepeating group or the "Entity A" to "Entity B" relationship group. Relationship and repeating groups, however, should carry a more expanded definition.

The element definitions are much more comprehensive. These definitions contain all the logical physical attributes, the field labels and headings, the edit values, if any, and other pertinent data.

NOTE: This is an iterative process. You will not get the structure right the first time. Do not worry about it. The structure is designed for ease of change. The objective is to get something up and running; do not waste your time designing data structures for days. Take a cut; then implement it; the physical structure is dynamic.

PHYSICAL DATA MODEL ANALYSIS

Data Base Administration is the generic term used in the industry to describe the Physical Data Model Analysis. Physical Data Model Analysis includes the conversion of the logical data model to a physical structure, and the implementation of that structure.

The first concern of the physical data model is, of course, the implementation of the Logical Data Model (i.e, the Entity Relationship Diagram and Logical Data Structure Diagram). The physical data model is the representation of how the data is actually stored by the Data Base Management System, in this case IMAGE. The first step of the conversion is to develop a Physical Data Structure Diagram which illustrates on a single page the application's logical view of the physical records (sets) which are required to support the application.

In this methodology there are two methods of physically implementing the logical data structure diagram into a physical data structure diagram. The two alternative implementations under this methology are:

1. Each logical data group in the Logical Data Structure Diagram can be given a physical record (set). One or multiple IMAGE data bases may be used. (See Figure 4)

2. All like logical data records can be stored in the same physical records (sets). This approach essentially places everything (all applications/entities) into one physical IMAGE data base. (See Figure 5)

The essential key to achieving a successful physical implementation of this approach is access by entity. Careful evaluation of the implementation within IMAGE and the conceptual approach to physical implementation is required to achieve the entity access. It may be generally stated that:

1. If you elect to use approach "1", this objective will probably occur rather easily. Keyed access to the non-repeating group of the entity will achieve access by entity.

2. If approach "2" is used, an additional entity of "entity" will have to be created to handle access. The "entity" entity" may be created within the logical data structure.

Rules of Thumb:

1. Stick with the normalized design. This will be your salvation for the future. It is the most flexible data structure and changes to it are easy (this is the crux of relational structures).

2. Normalize one step further. Create surrogate (system generated) keys for all entities and relationships with repeating groups. Create system generated keys for all relationships and repeating groups. This will add some more I/O but it is worth the effort. It eliminates writing key conversion programs further down the road.

3. Establish access "keys" only for entities. This approach will give you all the direct access paths you will need.

4. Decide at the beginning how you are going to implement entity relationships physically. There are basically two options:

PHYSICAL ENTITY MODEL 1

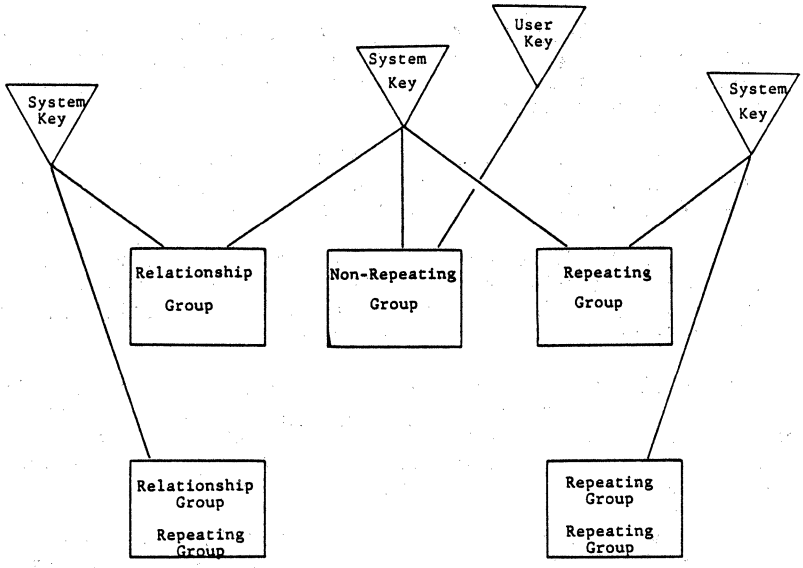


FIGURE 4

PHYSICAL ENTITY MODEL 2

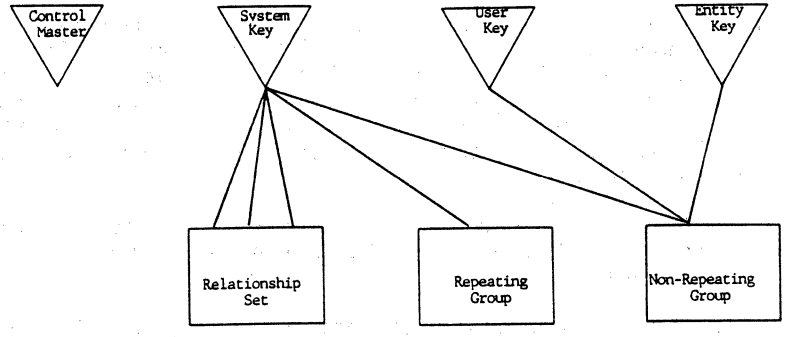


FIGURE 5

a. Create a relationship group for every relationship. This is undoubtedly the most flexible and will have far-reaching benefits in the maintenance of systems. It adds consistency to the actual method of chaining between entities and allows room for "policy" changes or logical data analysis errors when One to One or One to Many relationships turn out to be Many to Many relationships; or worse, have attributes.

b. Implement One to One and One to Many relationships as shown in the Logical Data Structure Diagram. This approach is more performance-oriented, and it is more understandable by the logical systems designer. It is not easy to fix if there is a change to the relationship. If there is ever the slightest doubt about the relationship, make it a relationship set; flexibility is the name of the game.

Note: Always work from the dictionary and the physical structures currently existing under this concept. If something exists, use it even (and especially) if it has to be modified to accommodate the current problem. You are simply overlaying one structure upon the other. Have faith in the design approach; it will allow you to do this.

The process required to implement the physical data structure depends upon which of the two methods of implementing the data bases is chosen. If method "1" (a physical data record for each data group), the process is as follows:

1. Using the Logical Data Structure Diagram (Figure 6 is an example of the Physical Data Structure Diagram for the Project Control System) as the starting point create a first cut Physical Data Structure Diagram. The first cut is an interim step and is only used for analysis purposes. The process is as follows:

   a. Establish a "key" record (Automatic Master Set) for each entity. This is the system generated key.

   b. Attach a record (detail set) for the non-repeating group to each key set.

   c. Add the relationship records and path structures. For each logical relationship group in the Logical Data Structure Diagram create a physical relationship group. (This is required for a Many to Many relationship and One to One and One to Many relationships which have attribute data modifying

Project Control System
Physical Data Structure Diagram

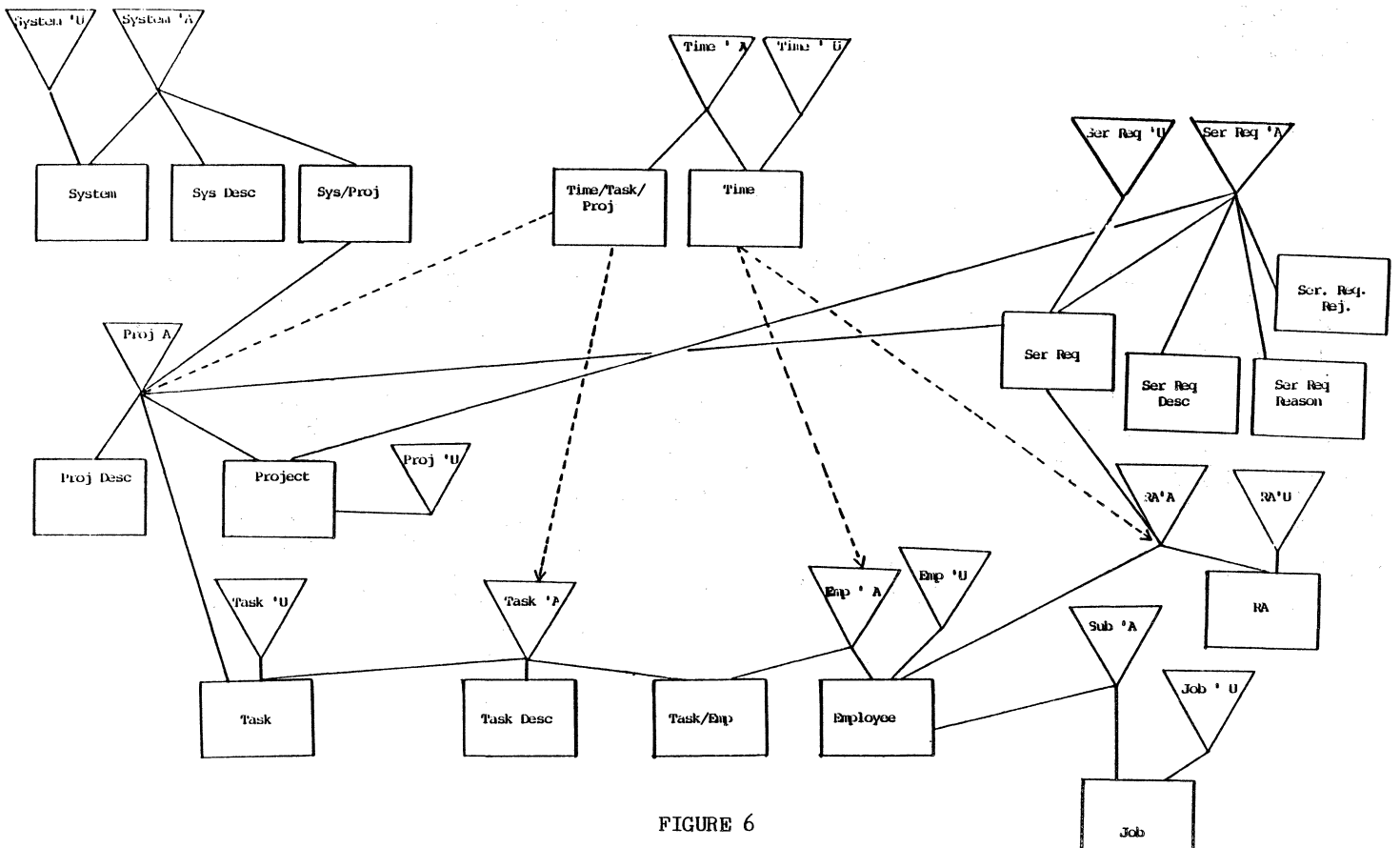


FIGURE 6

the relationship.) Create the relationship paths from the entity "key" sets to the non-repeating sets for the remaining relationships.

d. Add the repeating groups to the diagram linking them to the entity key (master set).

e. Add the relationship repeating groups and repeating group repeating groups to the diagram. Use another automatic master to link the two detail sets (Use system generated keys).

g. Under this approach, it is unlikely that the entire data structure will be contained in one physical data base. There will be physical and performance limitations. A method of creating links between physical data bases will be required. One method accomplishing this is to place duplicate entity key records (automatic master sets) in the secondary data bases. Choose your method for accomplishing this carefully.

Create the logical links between the physical data bases as needed.

2. Create the final Physical Data Structure Diagram. To accomplish this, there are only three further steps of normalization:

a. Add the user key access to the non-repeating group. These are the "user" access keys to the entity.

b. Review all the logical records. If any two are exactly alike, length and data items, place them in the same physical record chained to two entities. This process saves some disc space and will not affect performance (let the correlation between the logical and physical records be defined by the dictionary).

Once the Physical Data Structure Diagram is complete, enter it into the data dictionary and generate or modify the physical data base using the appropriate IMAGE utilities, dictionary utilities, or ADAGER.

If method "b" (a single physical data base) is used the process varies somewhat. Under this methodology, the physical structure is given and there is no physical design effort. The dictionary becomes the data base manager. In this situation the application code uses the dictionary to correlate calls for logical records with physical records. The discussion which follows concerns the concepts of the initial establishment of the physical data base.

Under this approach two levels of DBMS exist. The lowest level is the DBMS used to access the physical file, IMAGE. The second is the dictionary (Qschemac) which becomes a DBMS. In this approach, the application code (fourth generation report generator/query, screen processor, and transaction processor) interact with the dictionary. Calls are not made directly to the IMAGE. These languages access the logical files only by the name held in the data dictionary. The dictionary defines what the physical locaton of the record is and handles the redefinition of the physical records. This is an extremely basic concept. It may be compared to the old punch card where the "record ID" was a field in the card. This same principle is applied here. The record ID is the entity ID. Each occurrence of an entity is given a unique ID - unique within all occurrences of all entities. Each relationship is also given a unique ID within all occurrences of all relationships and entities. These ID's are system generated surrogate keys. User keys are used only to make the initial accesses to the non-repeating data set. The physical records established in this data base are:

1. **Control Master**. This set contains one record. The last system assigned surrogate key and is the control record for system generated keys. (This set is optional; date, time, and USER ID may also be used as the system generated key.)

2. **User Key Index**. This set contains user generated keys and links to the non-repeating set. This set allows direct read capability by entity occurrence.

3. **Surrogate Key Index**. This is the internal key file and links all the sets of each entity.

4. **Entity Key Index**. This set allows keyed access by entity. It provides a meas of serial access within an entity.

5. **Non-Repeating Group Set**. This record contains all the non-repeating group records for all entity occurrences.

6. **Repeating Group Set**. This record contains all the repeating group record for all entity occurrences, all relationship repeating groups, and all repeating group-repeating groups for all entities.

7. **Relationship Set**. This record contains all the relationship data between entities. All relationships are implemented as Many to Many.

This structure can be "tuned" for space by creating several non-repeating relationship and repeating group sets. Additional sets of "optimum" sizes (record length) may be created for the non-repeating and repeating group records. The

relationship set may be divided by the number of entity relationships; i.e., 2 entities or 3 entities, etc. If one of the existing records is not long enough or will not accomodate a 5-way relationship for example, an additional set can be added to the data base without impacting existing structures.

This type of structure can solve a number of chronic design problems. It can also cause a lot of problems. Recovery procedures are simplified; the length of time required to recover may, however, take substantially longer. Logging for audit purposes is also simplified and the problem of recovery for multiple data base updates is eliminated. Record volume and limitations of maximums of IMAGE could be problem areas. It is imperative that record level locking be used in this type of environment. Additionally, the DBMS must be able to control multiple batch and on-line simultaneous updates against the same data base. Recovery processes must also accommodate this processing environment. These requirements basically apply to both approaches.

Implementation of an application system's data structure merely requires relating the logical record with the physical record in the data dictionary (Qschemac).

The two approaches have advantages and disadvantages and the implementation will depend largely on individual perception of the capabilities and limitations of IMAGE and dictionary software.

The security concept is simple. Any user can read any data except for data items (fields) which have specified "user read access". Only specified users can write to a record. Security is defined in the data dictionary. IMAGE data bases are assigned passwords such that they can be accessed only through dictionary driven programs.

SUMMARY

The Entity Relationship Analysis process was derived from an analysis of large third normal form data bases and the processes which create them. This process of entity relationship analysis is a method producing basically the same data structure result at the physical file management level, but it uses a more intuitive and business oriented approach to the logical definition of the data.

The benefits of entity relationship analysis and the resulting data structures are:

1. Relational structures of operational systems can eliminate interface problems between applications and provide flexible data base which can better accommodate undefined future requirements. This is accomplished by overlaying application data needs over the previously existing data bases.

2. Analysis and design manhours are reduced. Entity relationships analysis is a proceduralized method, parts of which may be automated. Additionally, all output requirements of a system do not have to be defined in order to determine the data requirements and physical data structure requirements.

3. The process eliminates the "art" of data base design. Data base designs are either correct or incorrect. It provides a measurable "quality" attribute for performance analysis of design efforts.

4. The process provides a data structure which users and novice data base developers can easily grasp, even with large complicated data bases.

5. Training requirements for data base designers is a fraction of what is required for developing the abiltity to create good third normal form data base using normalization and data reduction concepts.

My experience indicates that relational data structures can easily be defined through entity relationship analysis. IMAGE can be used to accommodate these data structures with the help of some additional facilities. The key to relational data base is the implementation of data in third normal form. If data is so defined, then the data files can be migrated from a non-relational DBMS to a relational DBMS when that requirement occurs.

The process described in this paper has been developed in practical environments over the last six years and prototyped in production an electric utility data center for the last two years. It is currently being used to support 16 applications which include: Accounts Receivable, Land Resource Management, Master Equipment List, Maintenance Scheduling, Materials Management and Purchasing among others.

---

[1] ACM Transaction on Data Base Systems (IBM Systems, 1979), p. 397-434

Reference

Codd, Dr. E.F. "Extending the Data Base Relational Model to Capture More Meaning." ACM Transaction on Data Base Systems, Volume IV No. 4 (December 1979), 397-434

HP Distributed Systems Network Strategy for the 80's

Hewlett-Packard
Information Networks Division

19420 Homestead Road
Cupertino, Ca 95014

Douglas W. Tsui
February 18, 1983

HP Distributed Systems Network Strategy for the 80's

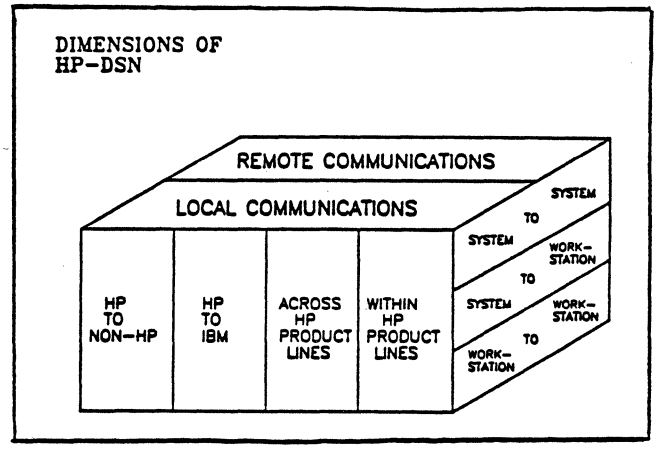
Hewlett-Packard
Information Networks Division

Douglas W. Tsui

This paper gives an overview of Hewlett-Packard's overall networking strategy for information management in the 1980's. The evolution towards a network solution for information management began with decentralized processing in the early 60's when data processing was done in the functional or geographical location where needed. With the advent of large mainframe computers in the mid-60's, the predominate thinking evolved into centralized processing where there is a greater centralized control and the computing cost can be shared among many users. Throughout the 70's, substantial advances were made in small computer technology. Distributed Processing was possible where minicomputers can be used to perform mainframe-like data processing tasks wherever it is needed or where it can be handled most effectively. Seperate computers are interconnected to facilitate data communication for information exchange. Now in the 80's, Hewlett-Packard is taking this Distributed Processing one step further. We are providing datacomm tools and products so that endusers can form their own integrated information management network consisting of not just HP computer systems, but workstations, mainframe computers, and other vendor's computers communicating both within a local or remote environment. It is a set of software and hardware that when viewed all together, provide for the exchange/transportation, storage, management, processing and access of information. An example of such an integrated solution is HP's **Manufacturers Productivity'Network (MPN).**

MPN is HP's computer strategy for manufacturing companies. It is a set of tools and applications to implement a network solution for information management in a manufacturing environment. In MPN, a company can be viewed in four major areas: There is the operational planning and control systems where an HP3000 system can handle distribution, cost accounting, materials management, production management, and order processing. There is the factory and plant automation area where an HP1000 system can perform process control, material handling, machine control, Computer Aided Testing, and automation control. In Computer Aided Engineering, HP has a broad line of desktop computers that can do Computer Aided Design, mechanical computer engineering as well as lab automation and engineering management. Finally, in the administrative and office service area, HP3000 systems and HP's professional workstation perform tasks such as work processing, personnel and payroll, document management, electronic mail, financial management, and decision support. All these applications are integrated together, providing quality and productivity through a linked solution. Linked solution are the key and that is the **HP Distributed System Network (DSN)**, the ability to provide an integrated network solution for information management.

HP DSN is an overall term describing HP's communication capabilities. This includes the whole set of hardware and software data communication products spanning multiple product lines to provide interconnection for HP's commercial systems, technical systems, workstations, as well as other vendor systems. These products can be directly and easily applied to solving distributed information processing problems. The goal is to provide communication products that form an invisible foundation for HP-supplied solutions and applications. At the same time, it provides enough flexibility to allow customers to establish or adapt these products into their own information network. In order to provide these kinds of capabilities, HP DSN can be viewed with having the following dimensions:



These dimensions allow us to examine all of HP's datacommunication capabilities in a logical fashion:

In communicating across and within HP's product lines, the emerging International Standard Organization (ISO's) Open Systems Interconnection (OSI) model will be used as the foundation. The OSI model provides a seven layer architecture with the enduser's application as the topmost layer. The presentation layer below gives any necessary translation, format conversion, or code conversion to put the information into a recognizable form. The session layer coordinates the interaction between the communicating end-applicaion processes. The transport layer assures end-to-end data integrity and provides for the required quality of service for exchanged information. The network layer governs the switching and routing of the information to establish a connection for the transparent delivery of the data. The data link layer handles the transfer of a unit of information between the ends of the physical link. The lowest level, the physical layer, provides for the transparent transmission of the bit streams to the connecting physical transmission media. It means that to send a message from one system to another, the message will go through all the layers in the architecture. Each of the layers adds a parameter that defines the function of that specific layer down through the physical layer until it forms a completed frame. The frame will then be sent across the transmission medium and the reverse process will take place. The parameters at each layer will be stripped and the specified function will be performed such that useful data will be finally presented to the application.

The benefits of such an architecture are twofold: (1) At the application level, an end-user will only need to learn about one user interface. As technology changes or new services are added in the lower layers to provide higher performance and more capabilities, the user interface will remain the same. (2) Specific HP systems can be selected to solve the application needs knowing that they can always communicate with each other. The end-user will only need to concentrate on solving the application problems with one set of user interfaces. Communication across or within HP systems will be assured.

In the remote communication area of the HP DSN dimension, access to packet switched networks, circuit switched networks, as well as dial-up and leased phone lines are provided. **CCITT X.25 Packet Switched Public Data Networks (PDN)** standard has been implemented on HP's DSN products. This allows system to system and system to workstation communication over PDNs such as Telenet in U.S. and Datapac in Canada. It provides a cost effective alternative beyond the traditional leased or dial-up telephone lines for digital computer communication. With PDNs, you only pay for the cost of the amount of data you send across the network and not the connection time. Multiple virtual circuits can be set up over one communication line for both system to system and system to workstation communication. Thus, it is a much better price/performance alternative for high volume, multiple connection applications. For an implementation of X.25 on the HP3000 system, please see the X.25 paper in this conference.

The X.25 standard is also compatible with the first three layers of the previously described ISO OSI model. Thus multiple vendor access across PDN is possible through this well-defined, standard user interface. In fact, HP's strategy for supporting multi-vendor networks is to provide user interfaces at increasing levels in the model between HP and other vendors that support the ISO standard. HP will provide tools and interfaces at each level above X.25 to allow endusers to implement multi-vendor networks based on the ISO architecture.

In the **Local Area Networking (LAN)** dimension, there is much interest in the industry these days. LAN can best be defined as a data communication system owned by a single organization within a limited geographical area. The benefit of such a system is to permit resource sharing within an organization and thus reduce the cost of Distributed Data Processing. It will also provide for sharing and exchanging of information and data as well. HP has been very active in such industry standard activities as the IEEE 802 LAN committee and intends to develop products that implements the standard, which is nearing full approval. Part of the standard defines a baseband, CSMA/CD (Carrier Sense Multiple-Access/ Collision Detection), 10Mbps, coaxial cable LAN.

However, HP also has committed itself to the use of Private Branch Exchanges (PBXs) as a principal communications method in its future computer networks for the office. Toward this end, HP has announced agreement to cover testing and certification of HP 3000 business computers and HP data terminals for interconntection to PBX vendors such as Rolm, Northern Telcom, and Intecom. HP sees no conflict with the two kinds of LAN technologies. In fact, we find them rather complementary. The CSMA/CD LAN is best suited for high-speed 'backbone' networks in applications such as system-to-system communication. In addition, they seem ideal for peripheral-sharing uses such as shared discs, printers and plotters. The PBX, on the other hand, is an excellent vehicle to handle large numbers of terminals and other slow to medium speed workstations. The twisted-wire distribution method and its integration with voice suit it especially well for office applications. The integration of voice and data in the office network provides better control, maintenance and diagnostics. HP expects neither PBXs nor CSMA/CD nets to dominate, but both to live compatibly together. As such, HP wants to provide each of our customers with the right network for the job.

In communicating with IBM or IBM compatible mainframes, HP is providing a set of products for batch Remote Job Entry such as IBM 2780/3780 emulation or HASP multileaving

workstation emulation and interactive terminal capability such as IBM 3270/3271 cluster controller emulation. For long term strategy, communication to mainframes will use IBM's System Network Architecture (SNA) as the base. This includes multi-function access to SNA networks via Physical Unit Type 2 (PU2) emulation. The objective is to easily adapt into our customer's information networks where IBM mainframes are used and to encourage IBM customers use of HP computers to solve their distributed information processing problem.

As you can see, HP DSN provides capabilities well beyond distributed data processing. It is a network solution with many dimensions to cover the need for an integrated information management system for a whole organization. It encompasses the OSI model for HP and other vendor communication. It provides access to Public Data Networks for remote communication and follows the industry standard in Local Area Networking. It supports multi-vendor PBXs. In all, it is the foundation for building a linked solution such as HP's Manufacturer's Productivity Network.

MPE PROGRAMMING

by Eugene Volokh,
YESOFT Consultants
506 N. Plymouth Blvd.
Los Angeles, CA 90004 USA
(213) 464-7523

Presentation to HPIUG 1983, Montreal, Canada

ABSTRACT

MPE, with its limited control structures and small set of commands, may not at first seem to be a powerful system programming tool. However, it turns out that MPE can be as powerful as (and easier to use than) any programming language for certain system programming tasks. This paper will try to introduce the reader, via a series of examples, to the art of "MPE programming".

THREE EXAMPLES OF MPE PROGRAMMING IN ACTION

Recently, in my capacity as systems consultant to a large HP installation, I encountered the following situation:

> There is a large system that can operate in one of two modes -- ONLINE or BATCH. Which mode it operates in is indicated by the presence or absence of a certain file called HOL100. If this file exists, this means that the system is operating in an ONLINE mode; if it does not, the system is operating in a BATCH mode.
>
> It is desirable to print at logon time which mode the system is running in.

Obviously, since something is to be done at logon time, we should use a logon UDC. The simplest solution is to have one of the form:

```
LOGONUDC
OPTION LOGON
RUN CHKSTAT
```

where CHKSTAT is a program that checks whether HOL100 exists and prints out an appropriate message. However, this is not the best solution. For one, I don't feel like writing a custom SPL program every time a rather simple systems programming task comes around; those who are not familiar with FOPEN will find this even harder to do. Furthermore, even if I did write a custom program for this, either the program or the source file is virtually guaranteed to get lost. And finally, running a program is a rather long and resource-consuming task.

But, if not a program, then what? After all, MPE does not even have a DISPLAY command to print a message, much less a command that will

check whether a file exists and display one message if it does and another if it doesn't.

At this point, I must make a confession; despite what I said of the possibilities of MPE as a systems programming language, it was by no means created to be a systems programming language. In fact, you will find that most of the techniques that will be described are actually methods of subverting MPE commands to do tasks that they were never intended to do in the first place. However, they work, and that's what counts.

Returning to the problem at hand, let us attack it one step at a time. For one, as I said, MPE does not provide us with a DISPLAY command. So, we'll make one!

UDCs are permitted to have a number of options. One of these options, LIST, instructs MPE to list out the commands in the UDC as they are executed. Furthermore, there is an MPE command called :COMMENT that does absolutely nothing. So, what do we get when we cross an OPTION LIST and a command that does nothing?

```
DISPLAY !STRING
OPTION LIST
COMMENT !STRING
```

When the above UDC is invoked via a command of the form 'DISPLAY "string"', it will execute the command 'COMMENT string' (which in and of itself will do nothing), but also list this command as it is being executed! Thus, if we don't mind seeing 'COMMENT' on the screen, we now have a way of displaying anything we want to on the terminal from within a UDC.

Thus, we've licked one of our problems -- we can now display a message to the terminal. However, this still does not solve the other problem -- determining whether a file exists or not and printing one message if it does and another if it doesn't.

Here, we must introduce a very important MPE construct (in fact, its only control structure) -- the :IF command. With the :IF command and its two sidekicks, :ELSE and :ENDIF, we can, depending on the value of a logical expression, execute one of two sets of commands.

Thus, our task can be expressed as follows:

```
LOGONUDC
OPTION LOGON
Check if HOL100 exists
IF it exists THEN
  DISPLAY "USING THE ONLINE SYSTEM"
ELSE
  DISPLAY "USING THE BATCH SYSTEM"
ENDIF
```

However, even before you start to furiously leaf through your MPE commands manual, you will probably begin to suspect that neither 'Check if HOL100 exists' nor 'it exists' is valid MPE syntax. In fact, there is no check-if-a-file-exists command in MPE. Or is there?

Well, if there is no command that will explicitly check whether a file exists, we ought to look for a command that, as a side effect, yields different results depending on whether a file exists or not. Furthermore, we would be able to differentiate these results using an :IF command.

Let us consider the :LISTF command. If we do a ':LISTF filename', the filename will be listed if the file exists, and a CI error 907 will be generated if it does not. Since we want as little output to the terminal as possible, we actually want to do a ':LISTF filename; $NULL', which will do nothing if the file exists, and print a CI error 907 if it does not. Furthermore, it turns out that the value of the last CI error is stored in a JCW (Job Control Word) called CIERROR, which can be interrogated via the :IF command. Thus, instead of 'Check if HOL100 exists' we should say ':LISTF HOL100; $NULL' and instead of 'it exists' we should say 'CIERROR<>907'. Thus, the solution to our problem is:

```
LOGONUDC
OPTION LOGON
SETJCW CIERROR=0
CONTINUE
LISTF HOL100;$NULL
IF CIERROR<>907 THEN
  DISPLAY "USING THE ONLINE SYSTEM"
ELSE
  DISPLAY "USING THE BATCH SYSTEM"
ENDIF
```

A few comments: 'SETJCW CIERROR=0' makes sure that CIERROR is cleared before the :LISTF command. Since this is an OPTION LOGON UDC, it is guaranteed to be zero anyway, but in general it is conceivable that it was already 907 before the :LISTF command. More importantly, a :CONTINUE command was added before the :LISTF command to avoid the UDC aborting on the first error; a :CONTINUE (either in a UDC or a job stream) instructs MPE not to abort if the next command fails.

One other point: in addition to the appropriate message, this method leaves some junk on the screen, namely the LISTF command and the error message if the file does not exist (and thus the LISTF command failed) and in either case a 'COMMENT' from the DISPLAY UDC. This is actually rather easy to take care of -- merely embed in the DISPLAY string some escape sequences to move the cursor and delete the unwanted lines and characters on the screen. If you're using printing terminals, though, you're out of luck.

Thus, we have seen how using MPE alone we can perform some fairly complex tasks easily and efficiently.

So, from this, we can derive a sort of MPE programming methodology:

1. If you see no direct way of performing a given task, try to find a way that yields the desired effect as a side effect, with little or no other direct effects or side effects.

2. If you wish to do two different things depending on some condition that can not be straightforwardly expressed with JCWs, try to find a command or sequence of commands that yields two different JCW values depending on the condition.

Let us take another example:

One of VESOFT's products, MPEX, is an extended MPE user interface that provides many desirable features, and is often 'lived in' by its users -- i.e. they run it once when they sign on, and stay in it until they are done, when they exit it and immediately sign off.

Some of our users decided to set up an option logon UDC of the form

```
MPEX
OPTION LOGON
RUN MPEX.PUB.VESOFT
BYE
```

This way, they would be automatically dropped into MPEX when they sign on, and will automatically be :BYEd off when they exit it. However, they do not want this to be done for jobs, but rather only for sessions. Thus, the task is to determine within a UDC whether one is in a job or a session.

In my opinion, in addition to the already existing system-defined JCWs such as JCW and CIERROR, HP should have provided us with JCWs such as MODE (to indicate whether we are a session or a job), FSERROR, etc. However, the fact remains that it did not, and we have to determine this for ourselves.

Let us apply our rule #2 -- is there a command that yields somewhat different results for job mode and session mode? In fact, there is. The :RESUME command, when executed from within session mode (but not from break mode, since the UDC will never be executed from within break mode) yields a CIWARN 1686 (COMMAND ONLY ALLOWED IN BREAK); however, when executed from within job mode, it issues a CIERR 978 (COMMAND NOT ALLOWED IN JOB MODE). Furthermore, since this is an OPTION LOGON UDC and will thus never be executed from break mode, the RESUME command has no other effects! Thus, our solution would be:

```
MPEX
OPTION LOGON
SETJCW CIERROR=0
CONTINUE
RESUME
IF CIERROR<>978 THEN
  RUN MPEX.PUB.YESOFT
  BYE
ENDIF
```

As an additional nicety, we may wish to do something like a 'DISPLAY "PLEASE IGNORE THE FOLLOWING MESSAGE"' before the RESUME command so that the user will not be puzzled by the warning that the RESUME command issues in session mode.

So, score another point for UDC programming.

To round out this section, consider one more example:

> Before performing a given task, we wish to find out whether a given file is in use or not. If it is not in use, we should perform the task; if it is in use, we should print an error message.

Solving this problem requires a substantial amount of knowledge file system. What we really want to do is to try to open the file with EXCLUSIVE, INPUT access; if the open succeeds, we want to close the file with SAVE disposition; if it fails, we want to set a flag.

However, we can not explicitly open and close files in MPE. Rather, we have to find a command to subvert so that it would do this task for us. This command's operation should be essentially similar to our target operation (i.e. it should do an open followed by a close). One command that pops to mind is the :PURGE command. Unfortunately, it opens a file with OUT access and closes it with DEL disposition.

But, via the :FILE command, we can force it to open and close the file with whatever options we please! Thus, our task may be achieved by doing the following:

```
FILE F=filename;EXC;ACC=IN;SAVE
SETJCW CIERROR=0
CONTINUE
PURGE *F
IF CIERROR=384 THEN
  DISPLAY "ERROR: FILE IS IN USE"
ELSE
  the file is not in use; do whatever is necessary
ENDIF
RESET F
```

Note that any open failure (except 'nonexistent file') during a :PURGE command causes a CIERR 384; furthermore, the last file system error is

not accessible as a JCW, so we have to assume that no other open failure will occur.

ADVANCED MPE PROGRAMMING

Consider the following problem:

> YESOFT distributes its products on a tape along with an installation job stream. When a user wishes to install the products, he :RESTOREs the job stream and streams it. The job stream creates the appropriate accounting structure, and then :RESTOREs all the relevant files off the installation tape. However, it is possible that some files can not be restored; in this case, we want to send an appropriate message to the console.

The obvious thing to do here would be to check CIERROR to see if :RESTORE failed, and if so, do a :TELLOP. But, :RESTORE does NOT set CIERROR if not all files were restored! It merely prints the filenames and the count of the files that were not restored to its list file, and terminates just like all files were restored.

We have run into a problem that we can't really solve with the techniques outlined above because no MPE command can examine the contents of a file for us. However, there is one HP utility that is made explicitly for examining the contents of files -- EDITOR!

Our plan of attack will be as follows: we will redirect the listing of the :STORE command to a disc file (by setting a file equation for SYSLIST), massage it with EDITOR, somehow cause EDITOR to set a JCW depending on the number of files not stored, and then, when we're back in MPE, examine that JCW.

So, our "program" will look like this:

```
:FILE SYSLIST,NEW; DEV=DISC; REC=-80,16,F,ASCII; NOCCTL; TEMP
:RESTORE *YESOFT; @.@.YESOFT, @.@.SECURITY; SHOW; OLDDATE
:RESET SYSLIST
:SETJCW FILESNOTRESTORED=0
:EDITOR
TEXT SYSLIST
LIST ALL
CHANGEQ "FILES NOT RESTORED",":SETJCW FILESNOTRESTORED" IN ALL
DELETEQ 1/*-1,*+1/LAST
KEEP $NEWPASS,UNNUMBERED
USE $OLDPASS
EXIT
:IF FILESNOTRESTORED<>0 THEN
:  TELLOP SOME FILES NOT RESTORED, CHECK SPOOL FILE!
:ENDIF
```

What does this mess do? Well, the first three lines do a :RESTORE, redirecting the listing to a disc file. Then, we enter EDITOR and

text in the list file. Now, we have to make EDITOR set a JCW depending on the number of files not restored. The way that we do this is by changing the 'FILES NOT RESTORED = xxx' line to ':SETJCW FILESNOTRESTORED = xxx' with the CHANGE statement, deleting all the other lines in the file, keeping this as a temporary file, and USEing this file! The USE command will read the file and execute the :SETJCW command that we put in it; now, when we exit EDITOR, the FILESNOTRESTORED JCW is equal to the number of files not restored, and can now be interrogated.

This kind of trick is a very valuable one, and should be added to out methodology:

3. If the parameters of an MPE command (in this case :SETJCW) depend on the result of another MPE command (in this case :STORE), redirect the listing of the latter into a disc file, and use EDITOR to create and execute the former. Similarly, if the input of a program depends on the result of another program or command, redirect the listing of the latter into a disc file, and use EDITOR to create the input file for the former.

This point is best explained by another example:

> VINIT, an HP utility, has a '>PDTRACK ldev' command, which prints the addresses of all the defective disc tracks on the disc device indicated by ldev. However, VINIT has no '>PDTRACK ALL' command. Implement it.

Applying our methodology, our strategy should be:

A. Find a command that lists all the disc devices in the system, and redirect its output to a disc file.

B. Using :EDITOR convert this output into input for VINIT.

C. Run VINIT using this newly-generated input file.

For step A, one command that seems to fit the bill is :DSTAT ALL. This little-known command produces output of the form:

LDEV-TYPE	STATUS	VOLUME (VOLUME SET-GEN)
1-7925	SYSTEM	MH7925U0
2-7925	SYSTEM	MH7925U1
3-7925	SYSTEM	MH7925U2

As you see, this command displays, among other things, the logical device numbers of all the discs in the system. However, one problem comes up immediately: unlike the :STORE command, whose output can easily be redirected to a disc file, :DSTAT ALL's output always goes to $STDLIST.

So, how are we to redirect the output of a command that can only send its output to $STDLIST? The answer is simple: redirect $STDLIST! Although we can not redirect the $STDLIST of a job or of a command, we can redirect the $STDLIST of a program. So, all we need to do is to issue the following commands:

```
:FILE LISTFILE,NEW; REC=-80,,F,ASCII; NOCCTL; TEMP
:RUN FCOPY.PUB.SYS; STDLIST=*LISTFILE
:DSTAT ALL
EXIT
:RESET LISTFILE
```

What we do is run FCOPY with its $STDLIST redirected to a disc file, and cause it to do a :DSTAT ALL. :DSTAT ALL will obediently print its output to $STDLIST, which has been redirected!

So, we have the :DSTAT ALL listing (along with some other stuff printed by FCOPY) in a temporary disc file called LISTFILE. Now, it is time for step 8 -- converting this :DSTAT ALL list file to a VINIT input file:

```
:FILE INFILE;TEMP
:EDITOR
TEXT LISTFILE
DELETE 1/6,LAST      << delete the various headers >>
FIND FIRST
WHILE
  FIND "-"                 << delete everything after the '-', >>
  DELETE *(*)/*(LAST)      << leaving only the ldev >>
CHANGE 1,"PDTRACK",ALL     << insert PDTRACKs before the ldevs >>
ADD                        << add an EXIT command >>
  EXIT
//
KEEP *INFILE
EXIT
```

We now have the VINIT input file; all we need to do is

```
:FILE INFILE,OLDTEMP :RUN PVINIT.PUB.SYS;STDIN=*INFILE
```

and we're done!

We finish off this section with one more example:

> VESOFT's installation stream signs on as MANAGER.SYS, builds the VESOFT and SECURITY accounts and streams two jobs, which sign on as MANAGER.VESOFT and MANAGER.SECURITY and build the VESOFT and SECURITY accounts. It is also the duty of the MANAGER.SYS job stream to restore the VESOFT and SECURITY files; however, it can not do this until the other two jobs finish. How can we make the MANAGER.SYS job stream wait for the others to terminate?

The key word in this problem is 'wait'. Again, on the surface it seems that MPE has no command that permits one to wait for a certain event to occur. Again, however, a trick exists that saves the day. This trick uses MESSAGE FILES.

Message files are a kind of file (introduced in MPE IV) that have the property that if a reader tries to read an empty message file, he does not get an immediate end of file, but rather suspends until the message file is no longer empty.

So, even before the two job streams are streamed, we build two message files in PUB.SYS: MSGVESOF and MSGSECUR. Furthermore, in each of the two internally streamed job streams, after we are all done, we write a record (via FCOPY) to the appropriate message file. And, in the main (MANAGER.SYS) job stream, right after we stream the two other job streams but before we do the :RESTORE, we read the two message files (again, via FCOPY). The resultant job stream goes like this:

```
!JOB MANAGER.SYS
!NEWACCT VESOFT
!NEWACCT SECURITY
!BUILD MSGVESOF
!RELEASE MSGVESOF          << so the job stream can write to it >>
!BUILD MSGSECUR
!RELEASE MSGSECUR
!STREAM ,#                 << stream the two other job streams >>
#JOB MANAGER.VESOFT
...
#FCOPY FROM;TO=MSGVESOF.PUB.SYS
VESOFT ACCOUNTING STRUCTURE BUILT!      << any message will do >>
#EOJ
#JOB MANAGER.SECURITY
...
#FCOPY FROM;TO=MSGSECUR.PUB.SYS
SECURITY ACCOUNTING STRUCTURE BUILT!
#EOJ
!FCOPY FROM=MSGVESOF;TO    << wait for the VESOFT stream >>
!FCOPY FROM=MSGSECUR;TO    << wait for the SECURITY stream >>
!RESTORE ...
!EOJ
```

The message file reads cause the job stream to suspend until the message files are non-empty, i.e. until the other job streams have written something to them. Thus, when the :RESTORE is executed, we are assured that the VESOFT and SECURITY accounting structures have been built.

CONCLUSION

I have presented some examples and some guidelines that should give the reader an idea of what MPE programming can do and how it can do

it. It is my belief that with this knowledge and some ingenuity, the reader can use the art of MPE programming to his advantage.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

BENCHMARK TECHNIQUES FOR CHARACTERIZING APPLICATION PERFORMANCE

David S. Wertheim
Hewlett-Packard

WHY BENCHMARK?

As the size of systems being designed today increases (and likewise the cost) more accurate information regarding the performance obtainable from these systems has become necessary. Having better data allows decision makers to make better decisions about the tradeoffs involved with adding additional equipment to an existing system, vs upgrading to a different system, or even just optimizing an existing system to perform better.

WHAT IS A BENCHMARK?

A benchmark is a test or series of tests designed to characterize the relative performance of two or more systems (or configurations). Benchmark is defined as "a point of reference from which measurements can be made".[1] The goal of a benchmark is to be able to make comparisons, based upon the most important measurement criteria. A very simple comparison can be made by comparing MIP's (millions of instructions per second) or even Whetstone scores of different machines. These techniques characterize the relative power of the CPU's and can position processors by raw horsepower. They are however not really benchmarks. A benchmark requires an application mix (preferably your own) that will produce results relevant to you. The most common benchmark measurement results are response times, transaction throughput rates (interactive) and elapsed times (batch). These measures best indicate performance in a way that is meaningful to the end user. For example, a typical transaction on a system might be to update an account balance after a payment is received. How many of these type transactions can this machine process per hour? Or, we have 50 data entry clerks doing "heads down" data entry using this particular software package, what is the average response time per terminal? What if I used a bigger (smaller) machine, what would it be then? How long does it take to execute this program on this system? Answers to these type questions are critical to making the right choices, and benchmarks can help supply the answers.

TYPES OF BENCHMARKS

Depending on the purpose of doing a benchmark, different types of benchmarks are available. The basic types of benchmarks are; 1) vendor vs vendor, 2) unit X vs unit Y of the same vendor, or 3) unit X configuration vs unit X with a different configuration of the same vendor.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

Vendor vs vendor benchmarks would be utilized when comparing price/performance ratios of different company's offerings. The customer is usually in a strong position here as all the companies vie for the customer's favor. The challenge in this type of benchmark is to measure the application EXACTLY as it will be implemented when it is "live", and to insure that all vendors are testing the equivalent solution. That is not to say they all have to choose the same implementation, but rather each can use their own implementation techniques as long as they stay within the same guidelines to accomplish the same results. All too often a tester will change a little something here, or modify a small requirement there. The best way to avoid these differences is by having adequate communication (via meetings), reviewing each vendors testplan, and interpretation of the benchmark proposal. This is particularly important when comparing the results. All output should be carefully reviewed to insure the specifications of the benchmark have been satisfied.

In many cases, the application is already written for one vendors machine, and the other candidates are required to convert/develop with respect to the original machine. Since this is a non-trivial task, be weary of the comment "we can simulate that type of application by doing such and such!" Usually this means we can generate the equivalent load of an interactive environment with a set of batch programs. Although this may be true for a single component (ie. number of disc I/O's generated) it surely is not true from a global system perspective. An operating system balances the delicate resource interaction between cpu, memory, and I/O demands, all working simultaneously. This can not be duplicated very easily by programs that only load specific pieces of the system. What can be said in favor of this "simulator" is that it is much simpler to construct and test. One must ask however, "What good is a simple test if the results have no bearing on the performance the end user will achieve when the machine is ultimately implemented?" The only way to insure THIS knowledge is to test the ACTUAL application!

Benchmarks of different units of the same vendor would be utilized when comparing price/performance ratios within the same company (ie. If I spend an additional X thousand dollars, what performance improvement can I expect?). This is a very common type of benchmark, alternatively stated "To upgrade, or not to upgrade." In order to accomplish this successfully, the tester must choose between "What is the effective change caused only by the change in CPU?" and "What is the best performance achievable for this total system?" Different techniques must be utilized in either case. Lets take for example a customer with a S/44, 4 MB main memory, 2 7925M and 1 7933H disc drives, a 7970E tape drive, and a 2680 laser printer. The customer is concerned because as terminals are being added to the system, performance is slowly degrading. The choices are to either test the same exact configuration - merely swapping a S/64 for the S/44, or of reconfiguring and testing this system: S/64, 8 MB main memory, 5 7925M, a 7976, and 2680.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

The factors that would affect this choice are; 1) understanding why performance is being degraded - is it a CPU, memory, or I/O bottleneck (or what combination), and 2) is the customer willing (or able) to change the system configuration, or are the bare bone changes necessary to improve performance desired? One consolation is at least the application remains constant between all these tests and therefore the independent variable is truly the change in system. (This will be one of the key points left to the tester to insure that this indeed is a true statement).

Multiple configurations of the same unit benchmarks can be used to optimize performance of existing systems, or of systems to be purchased. One must be very careful here not to get caught in "Test crazy mania." An acute condition where it is believed that every permutation of a configuration must be tested before a decision can be reached. Often more energy (and $) can be expended then the optimal solution can ever save or return. Success in this type of benchmark can be reached only when one variable is changed at a time. This requires extreme patience!

Many trends can be observed through a series of characteristic tests that represent improvements in memory, I/O, or operating system changes (ie. table sizes, etc.) The most remarkable of these type of tests have shed some tremendous light on the area of I/O performance and have led to conclusions about multiple master disc drives typically outperforming slave disc drives, and the 7920/7925 discs being faster than the 7933. Testing with additional memory is another simple test to see what performance improvements can be achieved. Some customers can remember the days when the C.E. made a house call with an additional memory board, installed it, ran the test and noted the performance improvements!

After over 75 benchmarks of the above types, some guidelines have been developed that can protect us from jumping to incorrect conclusions. The most important factor is merticulous consistency! If comparing different disc drives, make sure the systems are reloaded with the same file placement. The operating system should always be a constant factor (no updates between tests), as well as the table configurations. The tests should be run in exactly the same order! This is one that warrants further attention. At first glimpse, this seems like a trivial point, but experience has shown the order the tests are run DO have an impact on the results. Possibly the files that are built, therefore reside on different disc drives (MPE uses a round robin placement strategy) or even at different addresses on the same disc. This may not sound like much, but over a period of time, it can account for a significant difference. The rule is: keep as many variables constant as possible, and keep the testing implementation constant. If a counterintuitive result is found - investigate further. Most everything happens for a reason, and usually we can find the cause. There are still some "black holes", but by and large these investigations are the ones that lead to new discoveries. Finally, for a benchmark to be effective, adequate planning, testing, and documentation of results are necessary. An incomplete testplan leads to inconclusive results!

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

BATCH VS. INTERACTIVE

A benchmark is only as good as the application it measures! As previously mentioned, a batch benchmark may be simpler, but is not indicative of how an interactive application will perform. Many installations however are batch shops, so lets discuss how one goes about testing batch programs.

The main measure of batch program performance is elapsed time. Elapsed time is the time from program initiation to program completion. Sometimes this is also called wall time. Batch jobs are almost always tested stand alone, that is with no other load on the system. Variations include one single program, multiple programs running serially, or multiple programs running concurrently. In most cases running multiple programs concurrently has a shorter elapsed time than running the same programs serially, because the ability to overlap can occur. When running benchmarks of this type we suggest streaming the programs so that the output listings contain all the timings. One customer who had a S/III wanted to benchmark a S/64. Due to organizational growth, their serialized programs completed later and later, and finally were 3 hours behind when the information was needed (they could not use concurrent access because the output of each program was the input for the next program). The benchmark demonstrated with a S/64 they could complete their run with 2 hours to spare (a difference of 5 hours!)

While batch benchmarking is relatively straight forward, interactive benchmarking is complicated, involved, and cumbersome. In the old days interactive testing was done the brute force method. Line up X number terminals, with data entry clerks, and timers (to time a random cross - section of data enterers). From this, calculate average throughput rates and response times. Aside from some timing errors due to the reaction time of the timers (which will average out over time) this will work. Now add more memory and retest. Now update to a new operating system and retest. Now change a few key tables ... After having done this a number of times I can admit to you - it's no fun at all! (Think also of the opportunity cost of having 50 people testing instead of doing their jobs!) That is why most major vendors have gone to a technique called "Remote Terminal Emulation", or the ability to generate a workload on a system that looks and acts like a terminal workload.

The terminal emulation requires a minimum of 2 systems, a system under test (S.U.T.) and a driver system (see figure 1). The driver system formats inquiries and receives replies AS IF it were a group of terminals. This is transparent to the S.U.T., so it replies as if there WERE terminals at the other end. The driver system collects the responses and logs the time so it can calculate the response time, transaction turnaround time, and transaction throughput rates. Figure 2 illustrates the difference between these terms.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

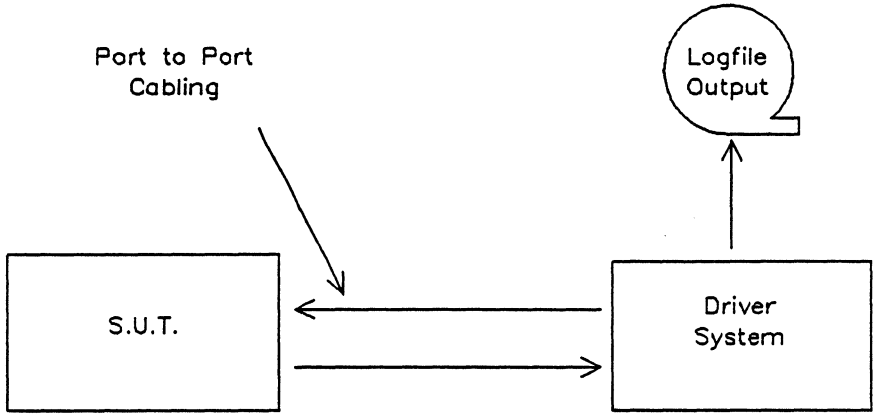


Figure 1

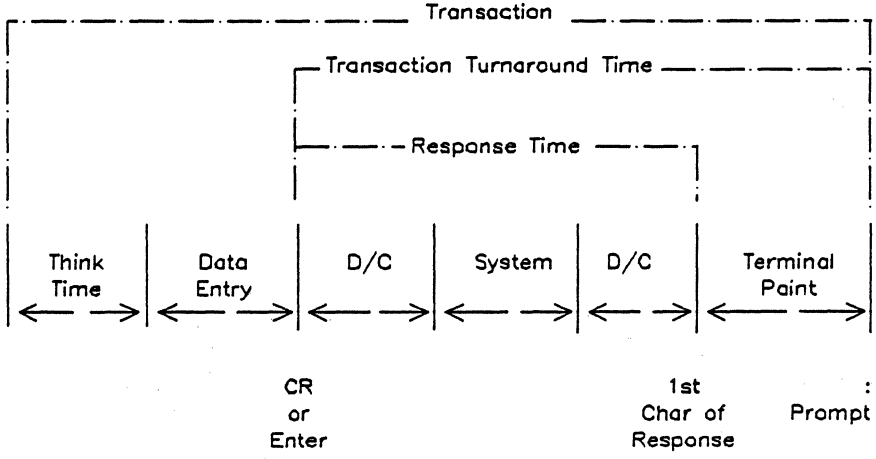


Figure 2

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

THINK TIME - The amount of time the user pauses before entering the next transaction. For data entry 0 to 5 secs, inquiry 5 to 180 secs, program development is random.

DATA ENTRY - The amount of time the user takes entering data. Typically .3 sec/char for data entry clerks.

DATACOMM (D/C) - Dependent on hardware utilized to communicate (each has a different character transmission rate).

SYSTEM - The time the system takes to process the request, CPU + Memory overhead + I/O subsystem (fetching data)

TERMINAL PAINT - The time it takes to fill the screen with data that is output from the system.

A transaction is a series of events beginning with think time, and ending when the system has returned control to the user (usually colon prompt or unlocked keyboard in block mode applications). The transaction rate is the rate with which the system can complete these transactions over a period of time. Response time is the time when a user enters the directive to the system to perform a task (or satisfy a request), until the system begins outputting the first character of the response. The reason for the destinction between response time and transaction turnaround time (see figure 2 again) is that the resulting output could be one character, or over 1000 characters. The measure called response time is the time the SYSTEM was occupied trying to respond to the request. Transaction turnaround time represents the time it REALLY takes to satisfy the request from the end users point of view.

The key to a successful interactive benchmark is to make sure the workload is representative of the true application mixture. A workload that accomplishes this is called a "characteristic workload". The closer the workload resembles the true application, the closer the benchmark results will reflect the live results that can be expected. The most important aspect here is to insure that the transactions used are those transactions that will be used on the system (and in the appropriate percentages). What percent should be data inquiry, data entry, MPE, program development, etc? How long does each user take to enter their requests (data entry time), and at what rate between requests (think time)? The answers to these questions will dictate what the results will look like. In this way, performance benchmarking is like statistics. By structuring the right parameters, a person can force results a certain direction. For example, if you desire a low transaction rate, merely have a very long think time. The very best transaction rate with 10 minute think time is less than 6 transactions per hour!

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

HOW DO I KNOW THE RIGHT PARAMETERS?

In actuality, the customer is the only one who can determine the validity of the parameters. The basic technique is to gather data about the users of the system and convert that to the appropriate specifications for the test. First, start with the number of users using the system. Can they be split out by project? How many per project? Within each project, can they be split out by function? How many per function? For each function, what is the typical profile of a user? That is, which transaction types does the user execute? How often over an 8 hour day? From this the transaction mix for the test will be generated. Each type of user now must be classified according to the way they use the terminal. How fast do they key in data? Observe them in action. For each type of user you will observe a bell curve distribution, slower -> average -> faster. We generally use the average data entry time for each user type. What is the think time profile of each user? Is it a constant amount of time, or is it randomized? When making these observations, be sure it is a typical workload on a typical day to avoid skewing the results. Often times we will use a think time distribution. Here we can specifiy what the bell curve SHOULD look like, and allow the emulator to generate the think times necessary to achieve that distribution. This whole process is greatly simplified if we take a one hour window and assume that is typical of the whole day. If this is unacceptable, we can generate characteristic workloads by shift, or by segment of the day, A.M. vs. P.M., etc.

HOW DOES HEWLETT-PACKARD DO THIS?

Trying to characterize performance by testing using live people was clearly an unacceptable solution, so TEPE (Terminal Emulator for Performance Evaluation) was developed. TEPE is a remote terminal emulator as previously described. The performance guides that are published characterizing system performance are produced with this tool.

To implement this requires a script. A script is the sequence of instructions that tells the S.U.T. what transactions to process, how many, how fast, etc. The script is what controls the benchmark. Each terminal to be emulated can have its own script, or it can share a script depending on the environment you are emulating. The major complaint we hear is "If I want to test 80 different terminals, does that mean I need to generate 80 unique scripts?" Not necessarily, but you do need to generate as many unique scripts as desired. Maybe your installation supports 8 groups, each of which has 10 users. If all 10 users were performing the same set of functions, this scenario would require only 8 different scripts. This is however a tedious and time consuming process. Some techniques to automate this process are now in their infancy and have tremendous potential. One technique is through software. Traps (or user exits) can be placed appropriately in application programs (or at the intrinsic level) to catch all requests (in this case, transactions).

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

These can then be logged to disc and later formatted for inclusion in the script. Thus the end user could merely be performing their normal work while the software collects their transaction profile. NOTE: The actual implementation is not as simple as this is made to sound, but nonetheless it can be done. Another technique is through hardware. For users with HP 2645's (with the cartridge tape capability) I have heard about a cable that can be built that will intercept the normal RS232 cable, and route all output from the terminal through the cartridge tape unit before transmitting to the system. The cartridge tape is later copied to the system and edited into the script. Again, caution about the general usability of this technique applies.

CAVEATS

To date, a hole exists regarding data communications and automated testing (remote terminals using MTS, the incorporation of DS, RJE, MRJE, and IMF). By in large, the only successful datacomm testing has revolved around characterizing the impact data communications has on a workload. The typical question answered is "What is the degradation when such and such is running? We plan in the near future to use a technique developed to test MTS, using the 2333A cluster controller to accomplish automated testing.

MODELS

No discussion about benchmarking would be complete without discussing modeling and how the two compare. A model is a program or set of programs that allow you to characterize relationships among variables. The objective is to be able to input queueing relationships (ie. number of I/O's per disc, or the arrival rate of processes in the cpu_queue) and output the response times and elapsed time estimates.

Historicaly, HP has avoided using models because the degree of confidence in the results of models has not reached the levels we desired. If after calculating the inputs, and running the model programs the confidence factor in the output is 75 - 80% sure, that was not typically sufficient. The emulation however produces results with extremely high confidence factors (we estimate 95 - 100%) since the application is actually running. As with emulation, the quality of the results is dependent on the quality of the inputs. The downside of emulation is the effort required to gain that level of confidence in the results (I'm sure you now have an appreciation for that). With the size of systems growing as they are, HP is moving in the direction of modeling, so we can satisfy the needs of the 80% of the community that only need that level of confidence, and reserving benchmarks for the 20% that must have acutely accurate results.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

In conclusion, we have examined different ways to collect data about application performance in different scenarios. Terminal emulation is very exact, but very cumbersome and time consuming. The key determinant of whether or not it is worth pursuing is the customers needs and commitment level. In order for this technique to be effective, the characteristic workload that is being emulated MUST reflect the application environment you wish to measure. If it does, the results will reflect real life achievable results. Don't try to estimate interactive performance with a batch benchmark - the results just won't correlate with the real world. When all this is put together properly - the result is a successful benchmark, with valid results that help the decision maker analyze the many tradeoffs and make a quality decision about the best alternative to pursue. Decision makers, I would like to leave you with this: Is any information better than no information? I used to believe that. Now I'd say, any ACCURATE information is better than no information!

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

BIBLIOGRAPHY AND FOOTNOTES

1. Svobodova, Liba, "Computer Performance Measurement and Evaluation Methods: Analysis and Applications", Elsevier, C 1976, pp 55.

2. Buzen, Jeffrey, "Tuning: Tools and Techniques", Infotech State of the Art Report on System Performance Tuning, June 1976.

3. Computer Decisions, "Capacity Management Fields Performance Dividends", May 1980.

4. Saunders, David, "Benchmarks Changing as Systems Mature", Computerworld, March 15, 1982.

5. Svobodova, Liba, "Computer System Measurability", Computer June 1976.

6. HP 3000 Computer Systems "Performance Guide",

7. TEPE (Terminal Emulator for Program Evaluation) Users Guide, January 1982.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

OPTIMIZING SYSTEM PERFORMANCE

David S. Wertheim
Hewlett-Packard

WHY TRY TO OPTIMIZE SYSTEM PERFORMANCE?

The most often heard cry from users today is bigger, better and faster! In a nutshell, they want to be able to do more work in less time. Many people from the old school say "throw iron at them".[1] That may be necessary, then again it may not be necessary. You wouldn't go to a Dr. who after you told them you had heart pains suggested open heart surgery would you? Would you? Often times fine tuning can improve performance of existing systems and satisfy user requirements. We will examine the most important factors that effect computer system performance and discuss measurement and evaluation techniques that will lead to reduced user response times, and improved system throughput.

Reducing user response times and improved system throughput seem noble enough goals, but these two goals have caused more grey hair and ulcers for MIS directors and system managers alike. Why? Because they are inverse relationships. If you want the very best response time - get a single terminal S/64. Whenever the user requests a task, all of the S/64's resources are at its disposal. The system throughput and price/performance suffers however, since the machine remains idle most of the time. Response time suffers when you try to optimize system throughput. I am always reminded of the image of lines at the supermarket. The maximum system throughput occurs when there are too few cashiers, and lines start to form. The cashiers are fully utilized (ie. always busy) but response time (the time to check out and pay) suffers greatly. In the supermarket though, you have the choice - to leave your basket and go to the nearest 7 - 11 and forget it. In a computer system you're not quite that fortunate (the break key does have some merit though). Our goal then is to find the right balance between acceptable response times and good system throughput.

I will discuss response time from the interactive point of view since online users are much more sensitive to delays in response. The batch user can merely substitute elapsed time for response time throughout this discussion. Response time is basically the time when a user requests a task, to the completion of that task.[2] The factors that determine that time are; data communications rate, system time, and terminal rates. I will leave datacomm and terminals for other discussions and will focus on the system component. This component can be split into 4 pieces; CPU, Memory, I/O, and the application. I will focus only on the first three, since the effects of applications on performance could fill many volumes, as we would need to delve into IMAGE, KSAM, COBOL, WORD, RAPID, graphics, Manufacturing software, Financial software ... [3]

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

The components we will discuss are important because the interaction of these resources control the length of response time. To improve response time, reduce this interaction time! In order to do this we must first understand how these resources effect response time, how their utilization can be measured, and finally how their utilization can be reduced. Throughout this discussion, percentages and guidelines will be indicated where possible. These are by no means the only "accepted" ones, but rather represent the opinion of the author.

CPU

The central processing unit (CPU) is the controller of the system. It processes tasks on a priority basis and can work on only one task at a time. The objective of effective resource utilization can best be met by utilizing the CPU and I/O resources concurrently. The profile of most transactions is depicted in figure 1.

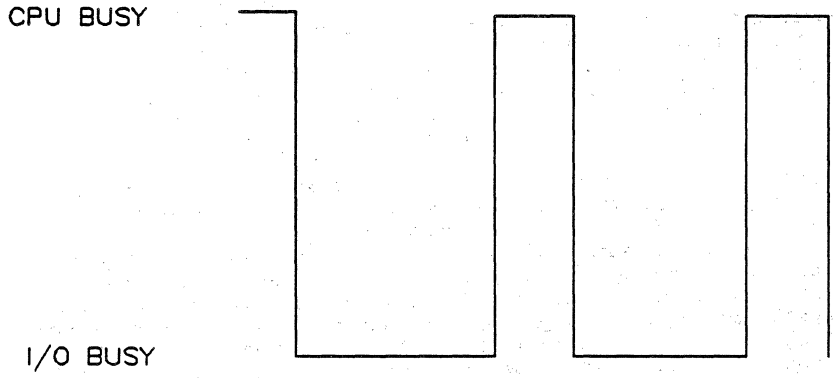


Figure 1

When the cpu schedules I/O for a process, it then becomes available for the next process which is "ready to run". If there are none, the cpu waits - until a process becomes ready, or the process executing I/O completes and it becomes ready to run. For a single user system, this would be like boiling a pot of water for tea, and standing there waiting for that event to occur - terribly wasteful. On the other hand, when many processes require the cpu simultaneously, most will have to wait for their time slice. Can you remember going to the bank, Friday the 31st of the month at 12:00 noon? Even though your transaction will only take 30 seconds, you wait in the line (queue) for your turn.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino California 95014 Telephone 408 725-8111

(NOTE: MPE does give you some flexibility by use of the scheduling queues to determine which processes get preferential treatment. For example, most users setup C Limit to have better priority than D Base Therefore any interactive request is satisfied before all batch requests. The only time a batch process will run is when there is no interactive process ready to run). We measure the effectiveness of cpu utilization by the percent of time a process makes a request, and finds the cpu busy with some other task (CPU busy %). This can be measured by OPT/3000 (On-line Performance Tool). A value of 80% or greater represents fairly heavy utilization.

HOW TO OPTIMIZE CPU UTILIZATION

In situations like these, the solution is to reduce contention for the CPU. This can be accomplished a number of different ways. Load management techniques can be implemented that would schedule heavy loads to non-peak timeframes (overnight, weekends). Another often used approach is to make database updates to a file that gets streamed during non-peak hours. This has the advantage of allowing maximum accessability to the database during high activity periods. The disadvantage is the database is not always in the most current state. Applicaton programs could be evaluated for redundant or sub-optimal code. This is an area that sometimes raises the neck hairs of programming staffs. You want to examine MY code? In many cases, tight scheduling has forced the elimination of the optimizing phase of the project. Since many applications proliferate to installations throughout the world, I am of the firm belief that optimizing that code once, will save an inordinate amount of execution time at end user sites. When you multiply that by the number of times the application will be run, the cumulative effect is staggering. (NOTE: This whole issue has received a lot of attention with the advent of programmer productivity aids. Typically these aids dramatically reduce the time needed to produce code. However, they also produce code that is not as efficient as an experienced programmer could.) APS/3000 (Application Program Sampler) can be used to isolate what segments (or even lines of code within segments) when optimized will leverage your efforts. Achieving 100% improvement in a segment that executes 1% of the time is not as good as a 1% improvement that executes 100% of the time! After these alternatives have been evaluated, the analysis sometimes indicates it would be more cost effective just to get a faster cpu to reduce the utilization.

This is not the case however if the utilization is under 30%. Additional horsepower will buy very little improvement. Referring back to figure 1, if the cpu component is 50 milliseconds, and the I/O component is 400 milliseconds (10 I/O's at 40 MSEC per I/O), what improvement can be expected if we double the cpu power - 25 milliseconds. By and large, this effect is negligible. As the old saying goes "all cpu's wait at the same speed!" Additional horsepower may complete the cpu component in half the time, yet while the process waits for other resources, the cpu also waits!

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

MEMORY MANAGEMENT

The effects of memory management on performance have been well documented so I won't spend much time here except for a few observations. With the price of memory dropping it does not make much sense to be underconfigured (some users with S/30/33's and S/II's are somewhat constrained). In fact, memory really only comes into play as a major component when it is underconfigured. In order to execute, a process requires that its stack, a code segment, and any extra data segments that it needs immediately be in memory. When a process references a segment that is not in memory (an absence trap), that segment must be brought into memory before the process can continue. If there is enough free memory, the memory manager allocates a free region for the segment and brings the segment in. If there is insufficient memory, a segment must be removed from memory (swapped out) or overlaid (only code segments or unmodified data segments can be overlaid without being written out to disc). This whole procedure is called swapping, and is controlled by the memory manager. When this situation is particularly aggravated, we call it thrashing. Thrashing could be defined as the memory manager busily swapping segments, but no useful work getting done. When a process is waiting for a swap to occur, and the cpu has no other process ready to run, we say the cpu is "paused for swap". A value of 10% or less is considered normal, 10 - 15% cautionary, and greater than 15% is serious. I'd like to make an interesting point on how numbers can be deceiving. I can guarantee a system would never show these percentages for paused for swap by running an idler program in the EQ that grinds cpu (perhaps a basic program: 10 GO TO 10). When memory pressure occurs and a process is waiting for a segment to be brought in (pause for swap would normally occur), the cpu would have a process ready to run and would therefore show this time as cpu busy time! I make this point because it does not have to be the idler that causes this, any heavy cpu utilization will demonstrate the same trend. Another indicator of memory pressure is the percent of disc I/O generated by the memory manager. When 25% or more of all I/O is generated by the memory manager, memory shortages are occuring. Much has been written about techniques to utilize less memory for those installations that are maxed out. These techniques range from shrinking the stack where possible, to removing static tables from the stack to code segments.

DISC I/O AS A BOTTLENECK

For today's systems the most important performance factor is clearly disc I/O. Discs are rotating mechanical devices and therefore operate at much slower speeds than cpus or memory. It is no wonder that disc I/O easily becomes a bottleneck. There are two ways to reduce this contention: complete each I/O faster, and request fewer I/O's!

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

Every disc I/O contains the following components.

	Set up Overhead	Seek	Latency	Transfer	Total Avg Access
7925	1.17	25.0MS	11.1MS	737KB/sec	37.27
7933	9.50	24.0MS	11.1MS	1060KB/sec	44.60

All discs have these pieces, and these speeds are the technological speeds available for discs today. When disc I/O requests are processed serially, the time to complete each I/O will be the average access times listed above. If multiple users request I/O, the sequence of events is depicted in figure 2. I use 40 MS as the average access time (the average of the 7925 and 7933 disc access times).

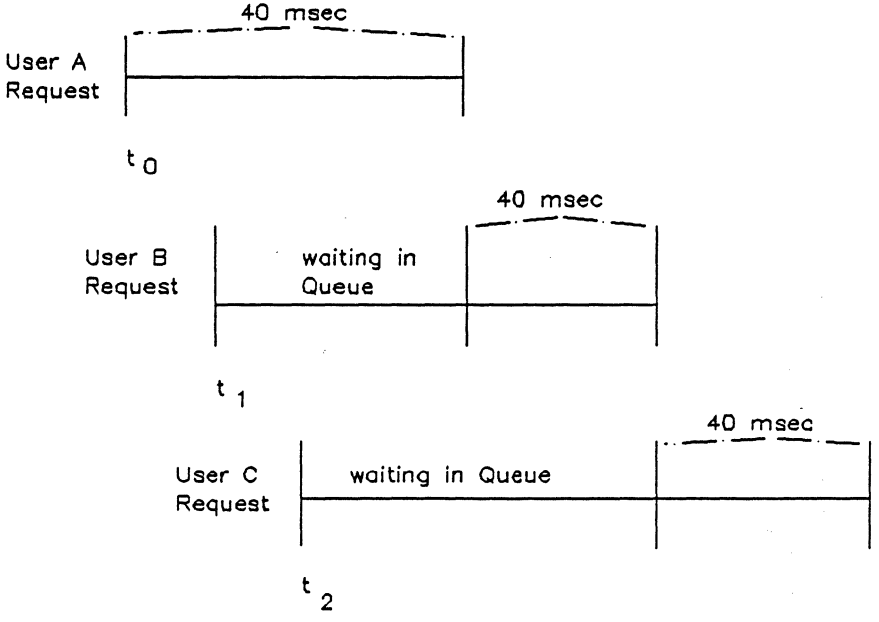


Figure 2

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

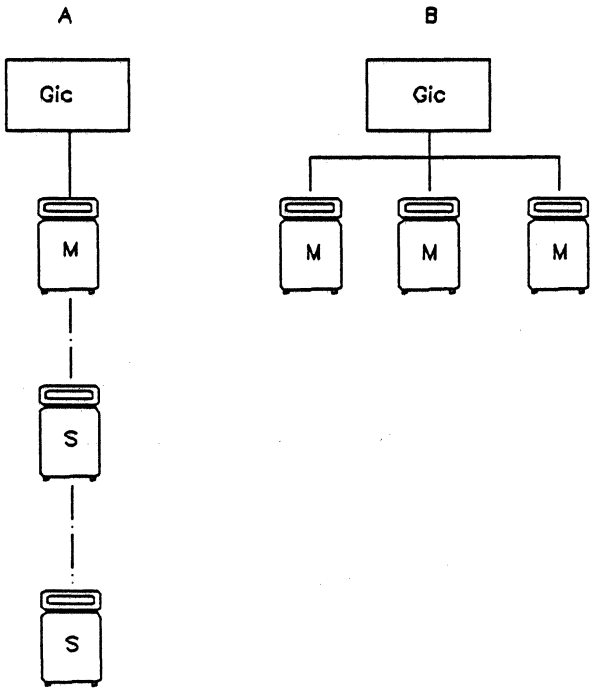
Notice that any user that makes a request while another user's request is being processed, waits until that request is completed before it is scheduled. The result then is the cumulative time! With MPE IV, changes were made to the I/O system that enable dramatic improvements in this area.

LOOK-AHEAD SEEKS

The S/III has a feature called look-ahead seeks. When enabled, this allows the user to have multiple seeks scheduled simultaneously, so up to 4 discs can be seeking at the same time. When the disc controller is being scheduled to seek, it checks the disc request queues for up to 4 discs to see if they can be scheduled at the same time. Once the seeks are completed, the transfer still remains serialized, but we've eliminated serialization of the longest component (the seek).

OVERLAPPED SEEKS

This same philosophy can be accomplished on the HPIB systems (S/30/33/39/40/44/64) by using master disc drives (a master disc has a disc controller attached to it). This technique is called overlapped seeks.



HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

The above diagram illustrates the difference between a master-slave configuration, and an all master disc drive configuration. System A's performance would be similar to that depicted in figure 2. While the controller is waiting for the I/O request to complete, all other requests are queued. System B however can schedule each disc drive independently since each has a controller. There is still the potential for a bottleneck at the GIC level, since only one controller can transfer at a time, however testing has shown with two masters/GIC negligible contention occurs, with slow rising contention from 3 to 4 masters/GIC, at which point the channel basically becomes saturated (see figure 3). That does not mean you cannot attach 5 or 6 masters/GIC, merely that there is less benefit of devices 5 and 6 being masters - they could be slaves and would perform close to as well. Figure 3 illustrates average I/O rates for the 7920/7925 disc drive.

CONFIGURATION	I/O'S / SEC	
1 M, 1 G, 1 I	22	
1 M, 2 G, 1 I	22	
2 M, 1 G, 1 I	44	
2 M, 2 G, 1 I	44	
3 M, 1 G, 1 I	53	
3 M, 2 G, 1 I	66	
4 M, 1 G, 1 I	60	
4 M, 2 G, 1 I	88	
5 M, 2 G, 1 I	94	
6 M, 2 G, 1 I	104	
4 M, 2 G, 2 I	88	ALL 2 IMB DATA IS
4 M, 4 G, 2 I	88	ESTIMATED! THE
5 M, 3 G, 2 I	110	SALES CENTER HAS
6 M, 3 G, 2 I	132	NOT TESTED THIS!

LEGEND:

M = 792X MASTER DISC DRIVE (10% less for 7933)
G = GENERAL I/O CHANNEL (GIC)
I = INTER-MODULE BUS (IMB)

Figure 3

HOW CAN THE USER CAPITALIZE ON THIS?

The best way to capitalize on these features is to spread requests across discs to gain concurrency of seeking, as well as a concurrency of I/O with CPU. In a single user system (batch job), no overlap can occur. If all users wish to access the same file, or a set of files that exist on the same disc - NO OVERLAP CAN OCCUR! How can a user determine this though? The best way I've been able to find is a set of contributed programs called "DISCIO". These programs analyze the log files looking for type 5 (File Close). They will format the file open/close activity based upon number of times this occured, and will also indicate the disc drive the file resides on. This will usually give a good indication of which files are most heavily utilized.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

A common rule of thumb is 80/20. 80% of all accesses are to 20% of the files (some people say with respect to disc files this is more like 93/7!). OPT/3000 will indicate which drives are most heavily utilized, and therefore that imbalances exist, but not which files. This philosophy of spreading accesses across discs pertains particularly well to virtual memory, as it flattens the disc drive distribution. (This used to be a big problem with heavy contention for LDEV 1 prior to MPE IV).

Another Strategy that seems to be effective is to split heavily used master and detail IMAGE datasets across different discs. If they are on the same disc, you are guaranteed a certain amount of head movement will occur to access a record. However, if they are split, there's a chance that subsequent accessess will not have to reposition the head each time. The amount of benefit this buys you depends on the magnitude and type of accessing being done. Finally, some have suggested placing your heavily used files toward the middle of the disc. The philosophy is, on average you will traverse half the disc per access. If you place these heavily used files in the middle of the disc, it will always be less than half the disc. In theory this sounds plausible, in reality it is not easily accomplished. MPE allows you to choose what disc drive, but not what cylinder or track. The previous suggestions do however work well, and please note: once you find an optimal placement strategy, back it up! Remember when reloading to use the RESTORE option, or you efforts will have been wasted. (For double protection map out your file placement strategy and keep it in a safe, accessible location!)

DISC I/O CACHING

When I first heard the disc I/O optimizing principle - request fewer I/O's I can remember chuckling, how absurd! If I could get by with fewer I/O's don't you think I would? That's the whole issue of adding memory isn't it, so there would be fewer I/O's necessitated by the memory manager? Yes, in actuality it is. Now we see much greater strides in thinking we call "Disc I/O Caching". The principle behind caching is to eliminate PHYSICAL I/O's and replace them by LOGICAL I/O's. Whenever an I/O request is made, it takes approximately 40 MSEC to complete. Of that time, only 1 MSEC/1000 bytes is for transferring data. By utilizing excess main memory as a cache for data with a high probability of access we can eliminate physical I/O's. When a physical access is made to the disc, instead of only transferring the requested data size, an additional amount is also transferred to the cache for future use. Clearly this will be very effective for serial access, however we have found even with random access locality does exist (the 80/20 rule again). What is the cost of this? The additional overhead necessary to transfer additional data (for 16KB this is 16 MSEC). The system can make a lot of mistakes (transfer data that turns out to be unneccesary) as long as it makes a couple of hits (the average cache access time is less than 5 MSEC/access). Preliminary testing has indicated greater than a 75% hit ratio! In order for this to work however, there must be adequate memory to utilize for this purpose. The cache regions are treated like any other memory region and are subject to being overlaid.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

If memory pressure exists, the hit ratio is likely to drop. Initial estimates indicate about 1 MB of main memory is necessary for this technique to be effective. To date, not enough testing has been completed to many any definitive statements.

HOW DOES DISC I/O CACHING AND MASTER DISC DRIVES JIVE?

The combination of disc I/O caching and master disc drives makes an interesting question. In other words, is it better to totally spread related data across master disc drives, or to bunch them on the same drive and capitalize on caching? Intuitively, my guess would be data that will be accessed simultaneously (or close to it) should be bunched together to allow the caching to be effective, since the access time savings is substantial. This could be done by overtly specifying the drive and using naming conventions like AA1, AA2, ... Since everything can't be bunched together, other related items should be spread across disc drives. This way, when physical accesses are required to replenish the cache, they will be completed expediently. In this way we accomplish both goals, complete each I/O faster, and request fewer I/O's!

One last subject should be addressed, that of resource contention for file locks, RINs, and other system tables. Both OPT/3000 and APS/3000 have the capability to isolate many of these deadlocks (TUNER2 for key system tables as well) and should be utilized to determine the causes of bottlenecks. A general rule is: hold resources for the minimum time necessary. This merely requires thinking through the way an application will use these resources from a global perspective.

In conclusion, techniques are available to measure the main system resources; CPU, memory, and I/O. Once the resource utilization is calibrated, bottlenecks can be isolated and steps take to alleviate contention. Through overlapping resource utilization, both system level goals can be accomplished; increasing system throughput and therefore reducing user response times. There is a balance between which of these requirements has higher priority, and that will determine the exact techniques utilized to accomplish that goal. In many cases, implementation of the above mentioned techniques can eliminate (or at least delay) the need for additional equipment in order to meet increasing end user demands.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

BIBLIOGRAPHY AND FOOTNOTES

1. Colloquial for "buy more hardware"

2. For a more detailed description of response times see "Benchmark Techniques for Characterizing Application Performance", David S. Wertheim (also presented at this conference).

3. A partial bibliography is listed below

Davison, Gerald, "Image Locking and Application Design", Journal of the HPGSUG.

Hulme, John, "System Design and Optimization Techniques and Tools", Journal HPGSUG, Third Quarter 1980, Vol 3, No. 3.

Rego, Alfredo, "Database Therapy: A Practioner's Experience".

White, Fred, "Improving Performance of IMAGE applications", Journal HPGSUG, May 1979, Vol 2, No. 4.

4. Greene, Robert, "MPE internals for Neophytes", Interact, July/ August 1982, Vol 2, issue 4.

5. Greene, Robert, "Optimizing On-line Programs", Technical Report, June 1981.

6. Howard, Jack, "System Design and Optimization Techniques and Tools", Journal HPGSUG Third Quarter 1980, Vol 3, No. 3.

7. May, Jim, "Programming for Performance", Journal HPIUG, July/ December 1982, Vol 5, No. 3,4.

8. Squires, Jim and Splinter, Ed, "System Performance Measurement and Optimization".

9. APS/3000 (Application Program Sampler) Reference Manual and User Guide, June 1982 (Part No.31280-90001)

10. OPT/3000 (On-line Performance Tool) Reference Manual, August 1981, Part No. 32238-90001.

Planning and Implementing an Automated Office

Patricia Wilcox
Hewlett-Packard Co.

What is Office Automation?

Michael Hammer, from Massachusetts Institute of Technology, has described an interesting perspective on offices and office automation. He asks that you think of the experience you might have had in explaining to a small child what it is that we do in offices. Anyone who has taken their child in to work with them has a joyous experience listening to the child tell what they saw or did: reading, talking on the telephone, shuffling papers, drinking coffee. What the child sees has very little to do with capturing the real "mission" of the business: making payments, scheduling production, etc. A view of the office describing the basic tasks gives very little understanding of the nature of office work. A number of observations can be made that help us understand what "office automation" or better yet, "business automation", is really about.

Offices and their activities are distributed both in space and time. They involve the continuous coordination of activities in many locations. Office activities are constantly changing. The worker must constantly revise his activities to cope with the dynamic environment in business today. Activities in the office are highly interactive. Most business decisions are made by the interaction of many people in an exchange of work, ideas, and commitment.

Hammer's perspective on automation reflects the "functional" approach to office automation. The office is a place of business, not just a place where document processing takes place. The real objective of OA is not the installation of automated systems into an office environment, but the enhancement of the business objective, using new technology only when and where it is appropriate. This perspective on "business automation" involves the integration of people, processes, technology, organization and change. The people (the human factor) are the most important part of the puzzle.

OA versus DP

The human factor has many side issues. In fact, that human factor is one of the main differences between office applications and data processing applications. The similarities seem more obvious, so let's review some of the differences.

Data Processing	Office Automation
Configure a technical system for precisely defined problem	System considers human factors
Automate manual procedures	Changes the way we work
Focus on needs of company & constraints of equipment	More tolerant of variety and types of users
Transaction processing	Supports decision making
Prescribed data and formats	Ad hoc analysis & inquiry
Oriented towards efficiency	Oriented towards effectiveness
Judged in terms of displaced costs	Judged in terms of value added

Office systems must be designed and implemented in such a way that they can measurably meet user needs and take into full consideration the human and organizational factors and constraints of the system. The design of such a system is critical as office automation affects every aspect of office life including:

1. the way of doing work
2. work flows
3. distribution of work
4. quality of relationships between workers
5. environmental design
6. "quality of work life"

The frequency and amount of human interaction in the office environment and the wide variety of situations requires a different methodology for systems implementation that most of us have experienced in the data processing environment.

Getting Started

There are a number of areas that need to be addressed as you examine whether or not office systems are appropriate for your organization. By asking yourself and your organization these five simple questions, you can be off and running:

Technological - Will it work?
Operational - Will it fit?

Behavioral - Will it be accepted?
Economic - Is it worth doing?
Regulatory - Are we allowed?

If you answered yes to those questions you can now start to look more deeply into the real issues and problems. To do that you must (terminology borrowed from IBM):

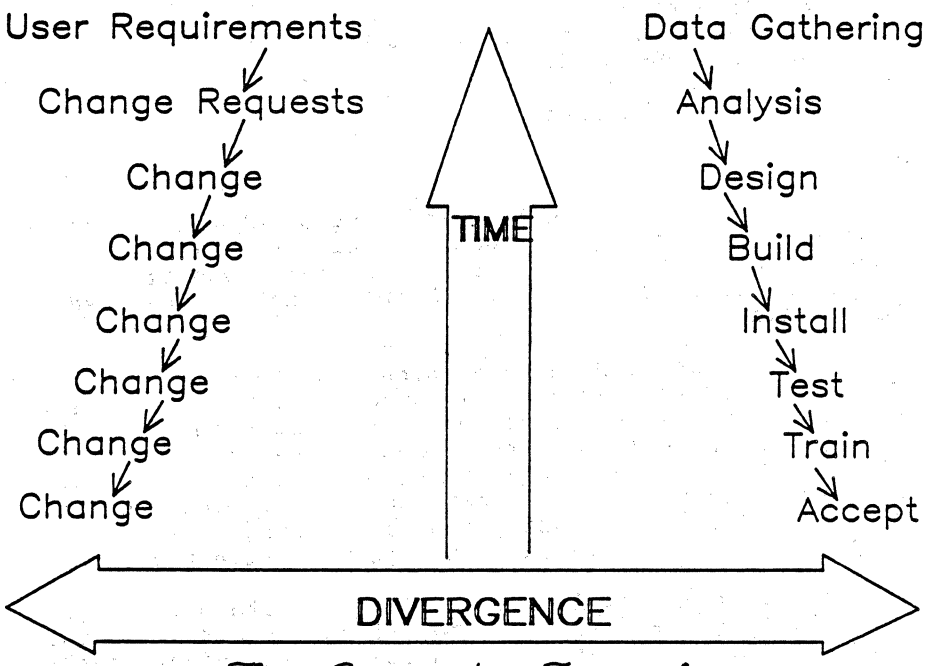
Organize to Plan
Define the Current Environment
Analyze Your Needs
Develop a Plan
Implement the Plan

Organize to Plan

The topics above, including the differences between the office and data processing environments, give some insight into one of the most critical part of planning and implementing an automated office. The user must be involved in the planning process.

The formation of the Office Automation task force is often the first step towards successful automation. In establishing this group it is important to get top management commitment early, and to establish the function and scope of the task force. Will it look only at the problem of office automation and make recommendations? Or will it function for a longer period of time, making specific plans, seeking their approval, and overseeing the implementation process?

From the experience that I have had working with office systems, the "User-driven" approach to office automation is the one that has the best chance for success. This user-driven perspective on office automation requires careful planning and education of the users along with continuing system review and refinement as the users become more experienced, their needs change and evolve. The "Bermuda Triangle" below illustrates the traditional implementation methods for computer systems where there is little user involvement. The "law of divergence" recognises that the office is dynamic. The longer it takes to build a system, the further it will be from the user requirements unless the user actively participates in the entire process.



The Bermuda Triangle

This user-driven perspective must take into account computer science, management, education, interior design, organizational development, industrial physchology, strategic planning and operations research and technology. In addition, there is a need to develop tools and procedures for office systems. All of this should be done with an evolutionary approach, melding the stragetic plan with practical experience.

User-developed office automation has two basic assumptions behind it: 1) all employees are responsible for improving office productivity, and 2) it's easier for office workers to understand technology than for technologists to understand how an office really functions. There are both organizational and individual benefits to this approach.

Define the Current Environment

Office systems can benefit an organization only if they address real business and employee needs, and accurately reflect the current environment. An inventory of existing office automation equipment must be taken. What commitments and choices have been made that limit your future alternatives? Remember that office automation is a merger of data processing, office processing and telecommunications. Each uses a different vehicle for information flow: digital data, paper and voice. You will need to collect information about these three information types and their flow to gain a total picture of your organizations requirements. The similarities and differences in departments are also an important aspect of your look at the current environment. A financial department may emphasize meticulous and repetitive work, while a sales department may be more concerned with the timeliness of the information and with customer service.

Analyze Your Needs

Needs analysis is a very important part of the planning and implementation of an automated office. Jan Duffy, a Canadian office automation consultant, has formulated the following needs analysis questions:

What are we doing now?
How much does it cost now?
How can the present system be improved?
What will the improvement cost?
What will be the impact on personnel?
How long is the process going to take?
What are the potential savings?
How will the new system benefit the organization in the future?

As part of your overall program the task force will need to address the cost-benefits issues of your proposal. There

are a couple of schools of thought on cost-benefit analysis. The first is that office systems prove their worth over time and there is no need to cost justify the equipment. Do you go through a cost benefit analysis for your telephone or your swivel chair? The second school of thought is that you should avoid doing anything. The third position is that you will find a way to justify what you already think. That brings us to Tapscott's first law of cost-justifying office systems: "the probability of a chooser accepting a cost-benefit analysis is directly proportional to degree to which s/he is favorably inclined to the technology anyway."

The benfits of office systems can be divided into both tangible and intangible benefits. An in-depth review of those should help you understand some of the trade-offs that you might have to make. These benfits translate directly to hard dollar savings and soft dollar savings. Hard dollar savings are things that either reduce the time required to accomplish the task and/or reduce costs of production. Soft dollar savings are the more intangible benefits such as improved quality and job enrichment. Tangible benefits generally save time, labor or both. Some examples include reduced need for travel and decreased number of phone calls. Intangible benefits are less susceptible to quantification. An example of an intangible benefit would be increased goodwill or customer satisfaction. A detailed look by application (electronic mail, decisions support) is important.

Your organization is undoubtedly going to be faced with the "productivity issue". I'm sure most of you are aware of the lack of well accepted productivity measures. There is also a lack of knowledge of the actual impact of office systems on performance and productivity. The best (and simplest) approach is to focus on the global measures of organizational performance rather than the efficiency and productivity of the individual in the office. One of the key contributions of the office task force is to help determine what productivity is for the organization. Is it quality? Employee satisfaction? Customer satisfaction? Efficiency and effectiveness. Can you describe the differences between efficiency and effectiveness? A look at some techniques for measurement and an idea of what a couple of large organizations did should be of some interest.

Develop a Plan

The office systems plan developed by the task force will the the bridge between your current office structure and the "office of the future". The plan should have a number of components. The first is a strategic framework. Where is the company going, what goals do we agree on and how are we going to get there. The organizational component should describe the methods of analysis and evaluation and how to prepare people for change. The process of adjusting to change has five steps:

Self-pity (Why me?)

Denial	(Not me!)
Anger	(I'll get even!)
Bargaining	(I'll cooperate if you will.)
Acceptance	(What change?)

Included in the plan should be a description of the office systems function. That included a mission statement for the function (jobs), the lines of reporting and working relationships, the definition of responsibilities and the staffing requirements. Along with the education and preparation for change, and the measurement issues mentioned above, the plan also needs to consider the implementation of computer technology and applications based on standards, guidelines and procedures to support the organizational objective. The plan also should address control of the process by management to ensure the completeness, cost effectiveness and responsiveness to the users.

Implementing the Plan

A pilot program is going to be a very effective method of introducing the organization to office sytems. There are a number of different types of pilots that are appropriate for the office systems environment. For example, you could choose a small pilot and use control and comparision groups for measurement techniques. You might choose an incremental pilot where your costs are minimized because you already have much of the equipment. Single application pilots, operational data base pilots, and combinations of all of the types mentioned can be sucessfully implemented. The key requirement for selecting the correct pilot is that it be located where there is a real need.

I'm sure, in the meantime, that you have been building an on-going relationship with your computer vendor. You need to balance your needs with the vendor services and strategic thrust. The office is, by definition, a "multi-vendor environment". You need to understand the compatibility issues and explain them to the users as part of the general education and training plan.

Have you determined who has what responsibilities during the implementation phase; the implementor, the user and the vendor? A clear deliniation of those responsibilities will help contribute to the success of the program. The attached Gantt chart should be useful in the planning and implementation process.

Evaluation of the Plan

During your pilot you should have been collecting and documenting users' reactions to the new system and recording their suggestions for system, procedure, training and other improvements. You need to refer back to your original criteria to measure the success of your installation and see

if they have been met. You must evaluate unforeseen advantages and disadvantages of the system. What was the effect of your training program, the vendor support and service, the documentation? What is the potential for additional applications? What effect does this have on the organizations' strategic plan. This is when you make the decisions to move from pilot to operational system.

Murphy's Law or the Pitfalls of Office Automation

What could possibly go wrong? Everything! A potential pitfall is the "technological imperative". "What is available and what can it do?" The team could also make long term decisions based on current technology. The team or management could confuse decision making with strategic planning. The project could reflect amateur planning. Unless the team has a background or training in strategic planning they will fall into one or both of the first "pits". Another potential problem is that the strategy might be developed on a one-shot basis as the day-to-day environment forces the team to delay or skip the planning process.

A review of other barriers to the success of the office automation project include organizational, people, implementors, and technological issues.

Summary

Successful office automation requires attention to all of the items mentioned above with particular emphasis on planning. In addition, the members of the team must:

1. give attention to detail
2. use checklists
3. have good people skills
4. excellent negotiating skills
5. budgeting skills
6. scheduling skills
7. education/training skills
8. communications skills
9. be resourceful
10. be open minded
11. be flexible

Foremost in the mind of the team needs to be the emphasis on the mission of the organization. The office is a system of interacting and integrated components (people, procedures and technology) that must work smoothly together to accomplish that business function.

In summary, if the solution fits the corporate strategy, considers the user's capabilities and needs, accounts for budget and time constraints, and uses reasonable assumptions about present and future technology and equipment; the result will be a comprehensive office automation program that realistically addresses short-term needs and establishes a frame work for satisfying long-term needs.

References

Don Tapscott; Office Automation: A User-Driven Method

Amy D. Wohl; Advanced Office Systems Concepts Corp.

Jan Duffy; Duffy, Bentley, Nelson-Turner Consulting, Ltd

Michael Hammer; Massachusetts Institute of Technology

Steven M. Abraham; Price Waterhouse, & Co.

"The Bermuda Triangle"; T.R. Conroy, Control Data Corp. &
J. Bieber, Contemporary Courseware

Arthur D. Little, Inc.

Hewlett-Packard
Office Systems

Gantt Chart

PLAN DATE _____
STATUS DATE _____
PAGE _____ OF _____

ACTIVITY _____

ACTIVITY/TASK	PRIMARY RESP.	PLANNED MAN-DAYS	
PLANNING			
Identify OA Survey Area			
- Review Company Profile			
- Conduct Office Systems Overview			
- Select Survery Area			
SURVEY			
Survey Preplanning			
Selection of Professional Interviewees			
Detailed Analysis Survey (Office Sys. Needs Analysis)			
- Briefings			
- Data Collection			
- Professional Interviews			
- Analysis/Summary			
Management Report/ Presentation			
EQUIPMENT/VENDOR SELECTION			
RFP Preparation			
Attend Vendor Demos			
Corporate Vendor Visits			

PREPARED BY _____

APPROVED BY _____

Hewlett-Packard
Office Systems
Gantt Chart

PLAN DATE _____
STATUS DATE _____
PAGE _____ OF _____

ACTIVITY _____

ACTIVITY/TASK	PRIMARY RESP.	PLANNED MAN-DAYS	
Vendor Selection			
FACILITIES			
Area Selection/Layout/ Design			
Construction/Remodeling/ Electrical			
Cable Requirements/ Installation			
Facility Completion			
Furnishing Order and Installation			
Equipment Order and Installation			
ORGANIZATION & STAFFING			
Define Organization/ Re-organization			
Develop Job Description & Rate Changes			
Hiring (if necessary)			
TRAINING & PROCEDURES			
Management Training			
End User Training Facilities/Instructors			
Communications Plan			
- Project Update to Management			
- Newsletters			

PREPARED BY _____

APPROVED BY _____

Hewlett-Packard
Office Systems
Gantt Chart

PLAN DATE _____
STATUS DATE _____
PAGE _____ OF _____

ACTIVITY _____

ACTIVITY/TASK	PRIMARY RESP.	PLANNED MAN-DAYS	
Equipment Demos and Films			
Procedures			
WP Operators			
Admin. Secretaries			
Professionals			
Skills Training			
Equipment Training			
System Orientation Sessions			
Work Measurement			
Management Reporting System			

PREPARED BY _____

APPROVED BY _____

Hewlett-Packard
Office Systems
Gantt Chart

PLAN DATE _____
STATUS DATE _____
PAGE _____ OF _____

ACTIVITY _____

ACTIVITY/TASK	PRIMARY RESP.	PLANNED MAN-DAYS	

PREPARED BY _____

APPROVED BY _____

"A CHECKLIST OF TEN FACTORS TO EVALUATE WHEN BUYING A SOFTWARE PACKAGE"

Thomas A.Wilson
Director, Profiles/3000 Sales

It used to be that you would buy some software, load it up, pay the vendor and then the real work for MIS would begin. With the common reductions in the price of hardware, there is more pressure on the software suppliers to differentiate themselves from each other. This means opportunity for the MIS manager in charge of purchasing software. Suddenly phrases like 'Let's try it out; Show me; Can you prove that' and 'Tell me about your support after the sale' have real meaning and can play a large role in the software that you ultimately select.

I feel that there are ten objective decisions for your consideration. Having completed that task you will have one subjective decision yet to make.

The following information if properly employed should lead you to a good solid selection that will have an ongoing positive impact on your productivity over the life of your selected software.

I. Documentation - Everyone talks about documentation and fortunately everyone is forced to do something about it. Unfortunately the normal function is to do as little about it as necessary. No documentation is going to fit specifically within your own standards, but some common questions that have been asked are:

1.) Does the documentation reside on-line for quick and easy accessability?
2.) Can I understand the documentation?
3.) How many kinds of documentation do you have? i.e. technical, user, training.
4.) May I see the user documentation?
5.) What does the training documentation look like? (This is in case I need to give additional training classes in the future.)
6.) What does the technical documentation look like?
7.) Is it understandable?
8.) Is it consistant with any of my other documentation?
9.) Can it be changed and if so, how easily?

As you can see there are many considerations from just one decision criteria. Without specific plan it is possible to make what appears to be a good decision but, find out later that it was an incomplete decision that led to disaster.

II. Training - This is a task that quite often falls on the shoulders on MIS. The ideal situation would be a training class given by the vendor with enough documentation that additional classes can be taught

by a representative from the end user organization. Some key questions for your consideration include:

1.) Is the training conducted on site?

2.) Is it classroom or terminal session oriented? (Tests show four hundred per cent better retention with terminal session classes.)

3.) Is the training documented for easy reference and also for on-going internal training? Is this documentation "user friendly" so that the layman can teach the class?

If your current policy is to not give internal training classes, you need to know if additional training classes are available and if so, what are the specific terms? Will the company come back to you or will you need to go to them? What are the charges for additional training and how long are those prices good? Is there a guaranteed response time to additional necessary training?

Training is a very important aspect in your decision because the software decision you make will be openly evaluated by the end user and judged on his/her ability to make the system do what they want. With incomplete or poor training an excellent piece of software will be given a miserable rating, and guess who will be blamed for this bad decision?

III. Service and Support - This area is probably the one that is undergoing the greatest amount of change in the shortest period of time. It is an area that is very costly for you and one where the most vendor differentiation is taking place. The functions to be dealt with include hot lines, trainer follow up, bug requests and product enhancement input.

1.) The most controversial consideration is the utilization of a hot line service. It raises the basic question of control, but properly implemented, can be controlled by MIS with a very high degree of increased productivity. End user problems have historically been the problem of MIS. Countless hours have been wasted researching non-problems. 85% of a users problems in the first six months are education and not system oriented. The good vendor will want to respond to these questions because he or she can spot trends and recognize necessary changes for future training classes. Also, patterns of questions in a multiple client relationship help develop easier and more efficient user instructions.

2.) All software contains bug request facilities in the maintenence agreement and this is an area to verify but is primarily a yes or no consideration.

3.) Enhancement input is a method of increasing your productivity. Find out the process undertaken by your proposed vendor and determine how much impact you may or may not have in this decision process. Some companies welcome your suggestion and given that they have a multiplier effect will implement them as maintenence. Others have internal departments that are not interested in outside input. Ask your vendor's policy and then ask for proof of this policy by verifying implemented changes and talking to references.

This entire process will be a potential area of increased productivity over the next few years. It is one that is now highly people intensive and requires a high level of end user knowledge. In all probability, this is a luxury that you will not be able to afford in the near future.

IV. Equipment Requirements - This is an area that you are probably most comfortable in but one that has a few pitfalls. Just because the software has an image database doesn't mean that it will fit your needs. You may be interested in whether it was designed specifically for the HP 3000 or perhaps it was converted from an IBM, Burroughs or some other system. You need to look at the design from an efficiency point of view. A natural question is; 'What are the minimum machine requirements and can I benchmark it's productivity?'

V. Installation Ease - After determining the efficiency of the software it stands to reason that you might be interested in how much work is involved in installing the package. No package is going to meet all of your musts and wants, so I would be especially curious as to the difficulty involved in modifying what I was buying and also what impact if any, that it had on the validity of my maintenence agreement. Additional information that will have a direct impact on the success of the installation include:

1.) Must all data be input intially in order for the system to function?

2.) Is a batch load facility available to initially load the data or must it be loaded item by item?

3.) Are there rules tiles established to insure the valid data or must all data be visually verified?

Another important aspect of the installation is the ability to identify, measure and control the entire process. Ask for proofs of timeframes and controls, discuss the company's claims in this area with some of the installed users.

VI. HP Interface - It is important to many companies that there is consistency in their software so the question of HP interface is important. The questions that readily come to mind are:

1.) Will this package interface with Rapid, Inform, Query and also, what about purchased packages like Quick and Quiz?
2.) Can I interface with word processors?
3.) If I hook up an HP micro, can I also use it as a terminal with this software?
4.) What are the requirements and limitations of terminal selection with this package?

VII. Maintenence - What do I get and what does it cost? Not bad things to determine early on in your selection process. As was mentioned earlier, the phone support issue is critical and becoming more so as people costs go up. What is the company's policy around fixing bugs, and if they are fixed do I get both source and object code on the fix? What about on-going enhancements? Am I going to get on-going improvements or am I caught up on the nickel and dime treadmill of continuous additional modules? This specification can have an impact on the true actual cost of your system.

Ask the vendor to show you proof of maintenence. This should be easy to do and will satisfy any apprehension you may have surrounding maintenence as an issue.

VIII. Security - All systems will have some form of password security and all security can be broken. Most end users will not have the ability to break any security controls so my suggestion is to check on additional security features and their associated benefits. Things to look for include levels of password security and the ability to institute securtiy to the data item level it desired. One benefit of higher levels of security

is that it can be controlled by the account supervisor and changed without having to involve MIS to unload and reload the database.

IX. End User Understanding - As more end users start using the computer, communication becomes critical. The terminology used by end users can be as frightening to the MIS department as the data processing terminology is to the end user. It is for those reasons that you want to determine not only the technical level of expertise of your preferred vendor, but also their understanding of the ultimate users needs. This should include not only knowing what and why the system needs to have certain features but also what will the needs be in the future. Has the system been developed with those future needs in mind and considerations made to accomodate those enhancements? If this vendor has passed your preliminary screens, it won't take long with the end user to see if they can communicate effectively.

It would not be out of line for you to ask for verification of this company's credentials in the expertise desired by your end user.

X. Vendor Viability - This is the area that in many cases is the only area scrutinized. I contend that the important aspects of this area are as follows:

1.) How old is the company you are dealing with?

2.) What is the financial stability of the company I am dealing with?
(This is critical in today's economy. What happens if the company goes out of business and what is the potential probability of that happening?)

3.) What is the geographic presence of this vendor? Do they have ten people or a thousand, and are they all located in one geographic area or are they accessable in many locations?

4.) What is the company's reputation? Do they follow through on commitments? What about other product lines? Can I verify the company's commitment to my needs?

5.) Will the company steer me to one or two special references or can I get a list of clients and talk to whomever I wish?

6.) How long has the company dealt in the functional area of my needs? How much do they know about my business and my end users needs?

7.) How long have they dealt in my specific marketplace? Do they understand my business? How well do they know the HP 3000? Am I a small part of their software service and controlled by what IBM users want or am I afforded equal treatment and consideration as an HP user?

XI. The Subjective Emotional Decision - If two or three vendors happen to pass your screens and only minor criteria differenciate them, chances are you will chose the company whose representative has gained your confidence. As a matter of fact, most software selection is made based on the relationship and amount of trust built up between the sales person and the decision maker or chief recommender. This is not bad decision making on your part, particularly if you are aware of it before making your final decision. As in all business decisions trust is a major consideration. That being the case accept it and use it to your advantage.

I truly believe that if you consciously consider all of the factors listed above that you will do a better job of selection software for your organization and that you will enhance your potential for advancement by measureably increasing productivity.

Applications of Message Files and their Performance Characteristics

Mike Zufall
Hewlett-Packard Company

Message Files are a feature of the MPE file system introduced with MPE IV. They provide a secure, transparent method for Inter Process Communication (IPC) that is both easier to use and more efficient than either the MAIL or Program to Program (PTOP) facilities they replace. This paper will suggest ways in which different installations might use message files, and detail the performance characteristics of message files in general.

Background

For those readers not familiar with Message Files, I will start with a brief explanation of the capabilities Message Files offer.

Message Files, first of all, are disc based and may reside either in the permanent or temporary domains. With a single process accessing a Message File, the file acts very much like a standard disc file. Records may be written to or read from the file using the standard MPE intrinsics FWRITE and FREAD. Two differences to note though are: a Message File may not be opened for input/output access; and, under default conditions, an FREAD deletes the record it accesses.

The real power of Message Files comes from accessing the file in read mode by one process, while one or more other processes are writing into the file. In this mode the Message File acts as a queue. It maintains both an end of file mark and a beginning of file mark (pointer). This allows the application designer to pass transactions or other information quickly, and easily between totally separate processes.

It should be noted that the above describes the standard ways of using message files. There are, in fact, many additional options that allow a great deal of flexibility. Using the AOPTIONS in the call to FOPEN, or a COPY parameter on a file equation, one may read a message file non-destructively; very useful in debugging. Using the AOPTIONS again, or a SHR parameter on a file equation will allow both multiple writers and readers.

A new (with CIPER MIT) and interesting feature of Message Files is Software Interrupts. This feature allows a process in, say, a compute bound loop to trap into a procedure of it's choice when a record is written into a Message File it

is reading. This feature works in a manner very similar to the control-Y subsystem break.

For the actual details of these and all other features of Message Files, the reader is directed to the references listed at the end of this paper. I will now procede to list some of the many ways Message Files may be of use to HP-3000 users.

Suggested Uses of Message Files

1. As an Aid in Application Conversion

Many types of systems require, or make it desirable that an interactive system be structured as separate processes. Sometimes an application designer will choose this scheme so as to make the system more portable from one vendor's equipment to another. With this type of design each process will usually handle one transaction type or perhaps all transactions for a given data base. There is also usually a small program that acts as a "front end" for handling terminal I/O. This creates a need for some form of communcation between these processes.

In the past, the only form of IPC available on a single HP-3000 was the MAIL facility. This presented several difficulties:

a. The MAIL facility was very inefficient for transfering more than one word of information. Sending a message one way requires a data segment to be allocated and released. Each allocation causes a data segment fault, increasing the total execution time of the process dramatically. When a reply is needed this overhead is doubled.

b. Either all terminals being controlled had to be in the same process tree, or, each terminal had to have it's own process tree structure. When all terminals are in the same process tree it becomes cumbersome to allow them access to other applications on the machine. When each terminal must have it's own process tree, operating system resources (such as the Process Control Block) can be quickly consumed by even a small application.

Message Files provide an elegant solution to these problems. They are easy to use because their user interface is the MPE file system intrinsics (FOPEN, FREAD, etc.), and secure because ACCOUNT, GROUP, FILE level security applies to them also. The main difference between Message Files and the MAIL facility is that Message Files are a "one way" street. A process may either read or write to a given Message File, but not both. To complete the loop you must use two different message files. In the case of a transaction

processor process it will read incoming transactions from one file and write it's responses to the file associated with the process that sent the message. The transaction processor may identify which process sent the transaction either by data embeded in the transaction or by enabling the extended read mode.

Some of the advantages of this type of application structure are:

a. When one particular transaction type becomes heavily used, the system management can start another copy of it running to improve performance. Note this assumes that two copies of the same program can truly compete with each other, i.e. they are not bounded by data base or file locking.

b. When it is certain that only one copy of the program will be run, that program can open it's data base or files exclusively eliminating the need for locking. If the program has exclusive access to an IMAGE data base, it can enable output deferred mode. This is, by the way, the only way a single transaction can achieve CPU/IO overlap.

c. Message Files may easily be file equated across DS lines to allow great flexibility in application growth. Keep in mind the 80-20 rule still applies; 80 percent of a files access should come from the machine it resides on.

You should understand that the above advantages are heavily qualified. When designing an application from scratch it is still best to use the traditional HP-3000 design concepts of having terminal and file handling in the same program. Rely on subprograms to break the application task into manageable pieces. However, when converting existing applications from other machines, use message files to make the conversion proceed quickly.

2. As an Interface to System Programs

This will be of particular interest to software houses or other sophisticated users. Many times this type of user will want to automate some additional part of the console operator's job. A prime example of this is spooler control. There is unfortunately no programatic way to control spool files, however, by combining message files with another MPE IV feature we can achieve the same effect. The other feature is the ability to redirect $STDIN and $STDLIST on the RUN command. By either streaming a job or creating a son process of SPOOK the user can gain access to all of the features needed to delete or alter spoolfiles.

A word of caution is due here though. The programs interfaced to SPOOK (or any other system program such as LISTDIR2) must be coded with knowlege of the output formats used by SPOOK. Since these are intended to be human interfaces, there is no guarantee that these formats will stay the same between releases of the operating system. As additional features are added, such as class print capability, these formats will be certain to change. If your application depends on these formats, be ready to test and change with each new revision.

3. A Link With System Commands

Many times a user will wish to process a group of files based on one of their attributes, such as file code (KSAM, VFAST, etc.). Although there are some unsupported utilities that can do this, usually there is enough difference in the task at hand so that they won't work.

What the user is then left with is a time consuming LISTF into a disc file. The disc file is then processed by a small "quick and dirty" program perhaps written in BASIC. Unfortunately, many systems just don't have enough free disc space to handle this easily. This is especially true in a university environment where there tends to be many, many small files on disc.

Doing the LISTF to a Message File offers two advantages over a sequential disc file. One, a smaller amount of disc space can be used, and two, you can achieve overlap in the listing of the files and the processing of them thereby reducing the elapsed time to complete your job. All that needs to be done is to stream two jobs. The first contains the LISTF command and the second contains the processing program. The two jobs form a producer/consumer pair.

4. Communication Across a DS Line

As a replacement for PTOP intrinsics Message Files offer several advantages:

a. Testing and debugging can take place on the same system. Again, because Message Files use standard MPE file system intrinsics, the file can be reassigned at run time to be either local or remote.

b. Multiple versions of the same program can be eliminated. Most DS users are larger customers who many times have machines scattered over the world. Since Message Files can be reassigned at run time, those sites not having DS connections can use the same program versions as the other sites which do. This is assuming that DS communication is taking place with PTOP intrinsics.

c. Communication is more efficient. Message Files out perform PTOP in all cases. Remember though, that for the DS HP-3000 to HP-1000 connection PTOP is still the only supported method.

d. Communication may span through several systems. Because a remote file equation may be assigned to another remote file equation, communication may start at system A, go through system B, and end up at system C. This is increased functionality over the older PTOP method.

For all of the above reasons Message Files are now considered to be the preferred method of inter system/inter process communication.

5. Controlling Multiple Tasks

For users in a program development, or other environment that wish to interact with more than one process at a time, the combination of Message Files and redirected $STDIN/ $STDLIST provides for some interesting possibilities.

It is possible, for example, to construct a small program which provides several "logical channels", each of which can create and activate a separate process. The user interacts with the father process which sends the command input to the appropriate "logical channel". Soft interrupts can be used to insure that all son processes have an equal chance of output.

This concept can be expanded across DS lines to provide for a convenient method of operator control on remote systems.

6. Construction of Spooler Programs

Message Files can be an invaluable aid in the creation of specialized spooler programs. Many users interface foreign or custom peripherals to the HP-3000 through the use of asynchronous terminal ports. Often these devices must use terminal type 18 with the controlling program reading and writting protocol characters directly to the device. This precludes the use of standard system spoolers.

With Message Files the user now has a simple, direct method of communicating with the process controlling the peripheral from sessions and jobs across the system. Message Files eliminate the need to communicate through a data base coordinated by locking.

With Message Files the user may choose to route all output to the controller process, or only the notification that there is a file waiting it's turn to be output.

Performance Characteristics

The approach I have taken to detail the performance characteristics of Message Files is to compare them to standard

disc files, MAIL, and PTOP. For testing purposes I constructed several small programs which exercised one particular aspect of Message Files in an extreme manner. The performance of these small programs was then measured by using MPE Data Capture Program (MPEDCP) and MPE Data Reduction Program (MPEDRP). The stand alone measurements were taken on a dedicated Series 40 with 768K bytes of memory and two disc drives, a 7933 and a 7912.

1. Message Files vs Standard Disc Files

The common uses of Message Files are very different from that of standard disc files. However, since they are both disc based and both accessed with the same system intrinsics they do provide a good starting point for comparisons.

The tests comparing Message Files to standard disc files are done in three different areas. They are:

a. File opens and closes

b. FWRITES to the file

c. FREADS to the file

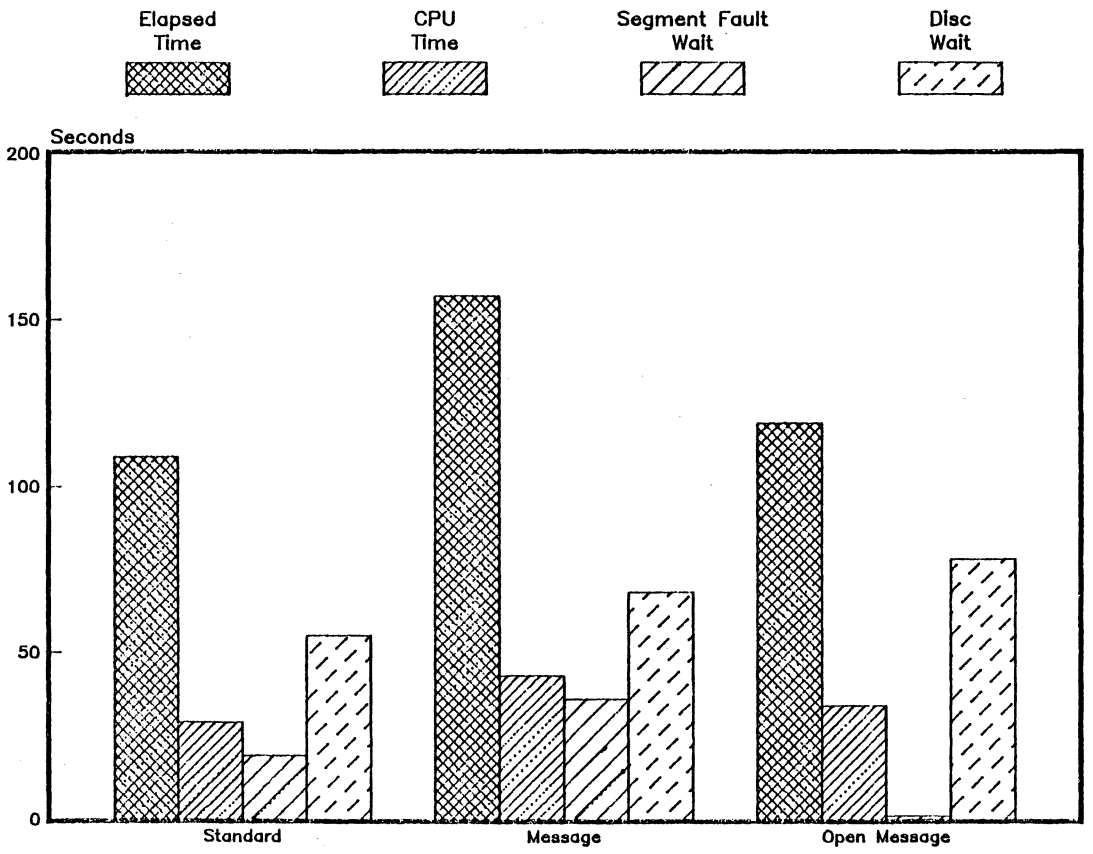
For the test of file opens and closes, a small SPL program was written that loops 500 times opening and closing a file with the actual designator "DISCFILE". "DISCFILE" was changed in between tests from a standard MPE disc file to a Message File. A simple BUILD command with only a parameter of ";MSG" was used.

For the tests of FWRITE and FREAD, a small SPL program was written that opened the file for read or write access, and then proceded to read or write 10,000 records to or from the file. For this test a BUILD command was issued as before, but an additional parameter of ";DISC=10000" was added.

The test results showing Elapsed Time, CPU Time, Segment Fault Wait Time, and Disc I/O Wait Time are shown on the next three pages in graphic format.

Message Files vs Standard

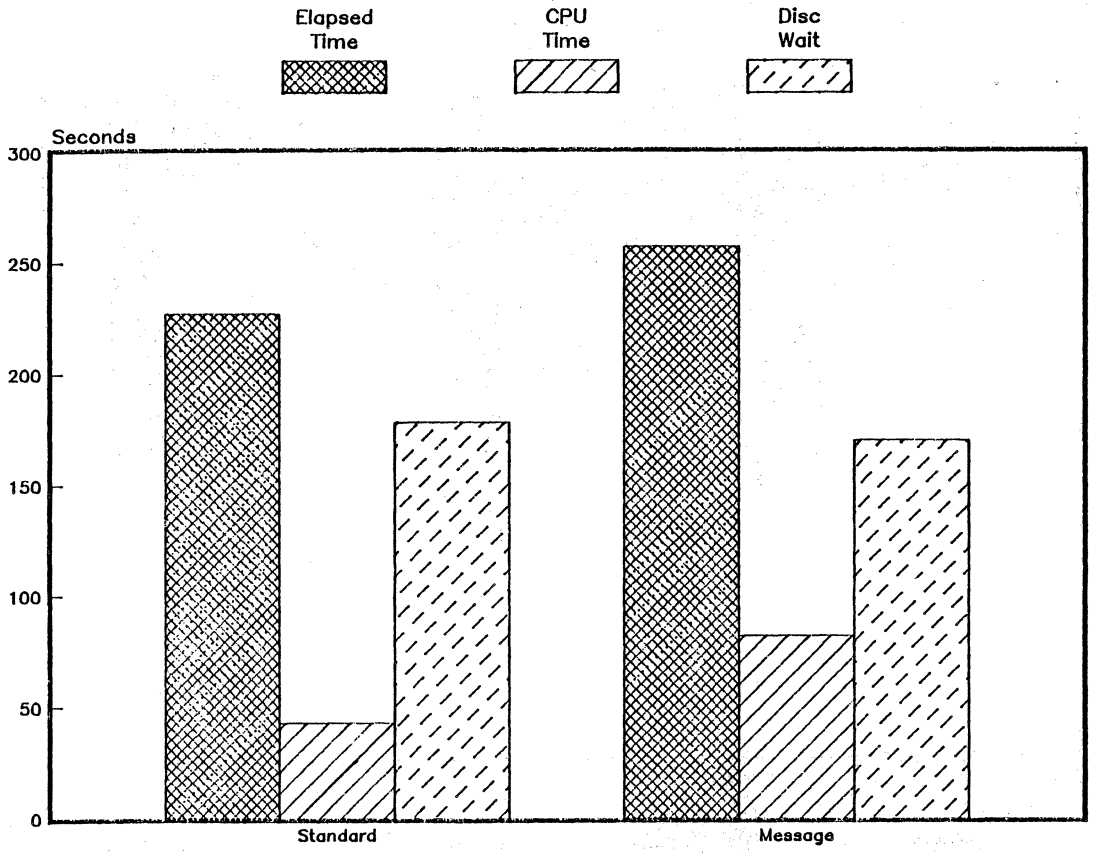
500 File Opens and Closes



Stand Alone Series 40 – C.00.20

Message Files vs Standard

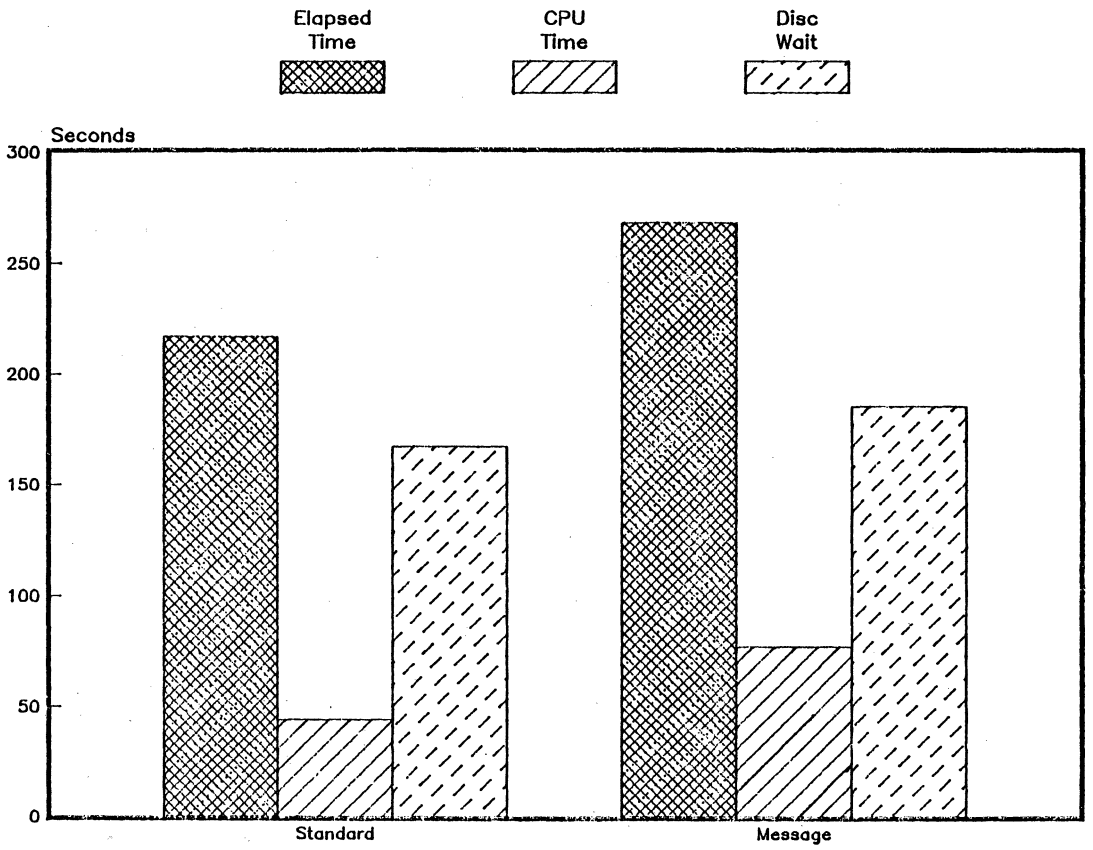
10,000 Reads



Stand Alone Series 40 – C.00.20

Message Files vs Standard

10,000 Writes



Test Type

Stand Alone Series 40 – C.00.20

Additional Comments and Conclusions

The additional complexity of Message Files require about 50 percent more CPU for the first FOPEN than a standard MPE disc file. The elapsed time, however, only increases by about 20 percent. Although not shown in the graphs, it should be noted that Message Files segment fault twice as much on the FOPEN/FCLOSE as do standard disc files. The reason for this is that Message Files require two data segments rather than just one. For the second FOPEN of a Message File, the CPU time and disc wait time increase slightly, but segment fault wait drops to almost zero because no new data segments need to be set up. The net result is an elapsed time very close to that of a standard disc file. On an average, the standard disc file open and close required five disc I/O's while the Message File open and close took six disc I/O's. This was true wether the Message File was open or not.

For the read and write tests, Message Files take more elapsed time than standard disc files. This is due almost totally to the increase in CPU time required to handle the more complex structure of Message Files. Both Message Files and standard disc files used one disc I/O for each read or write and spent the same amount of time waiting for disc I/O to complete.

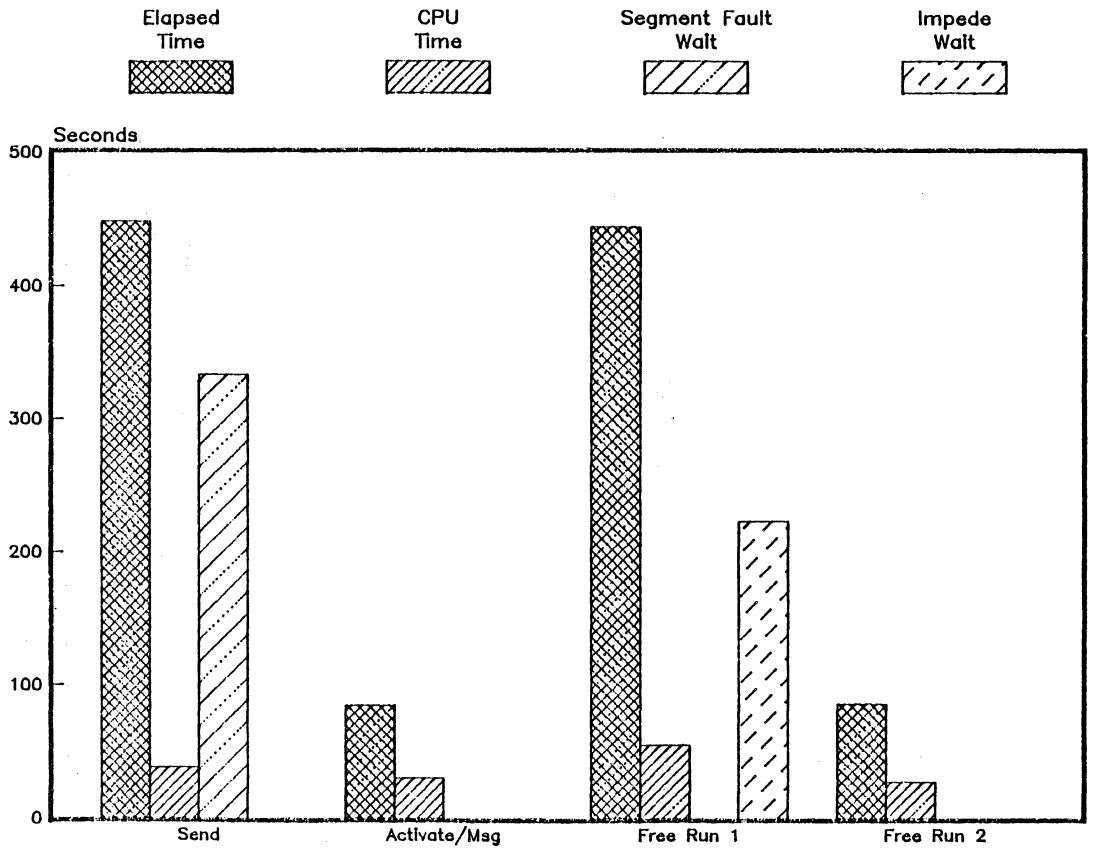
2. Message Files versus MAIL

Probably the most important use of Message Files is as a replacement for the MAIL facility. To characterize performance in this case, a total of eight tests were set up, four for the SENDMAIL/FWRITE combination, and four for the RECEIVEMAIL/FREAD combination.

For these tests two small SPL programs were written, one acted as a father process, and the other acted as a son. For the MAIL tests the father process created the son, then sent a 128 word buffer and activated the son while at the same time suspending itself. The son received the message then activated the father process again. In this way the activate with suspend coordinated access to the MAIL box. The first test of Message Files contained the same type of activate calls, but replaced the sends and receives with writes and reads to a Message File. The second test of Message Files removed the activate calls allowing the two processes to run at their own rates. Note that the father process does the first write to the file. In both of the previous tests all defaults were taken on the BUILD of the Message File. The third test of Message Files let both processes run at their own rates, but changed the BUILD of the Message File to have two records per block and a capacity of 30. At run time, a file equation was used specifying 16 buffers. The results of these tests are shown on the following two pages.

Message Files vs MAIL

10,000 Sends/Writes

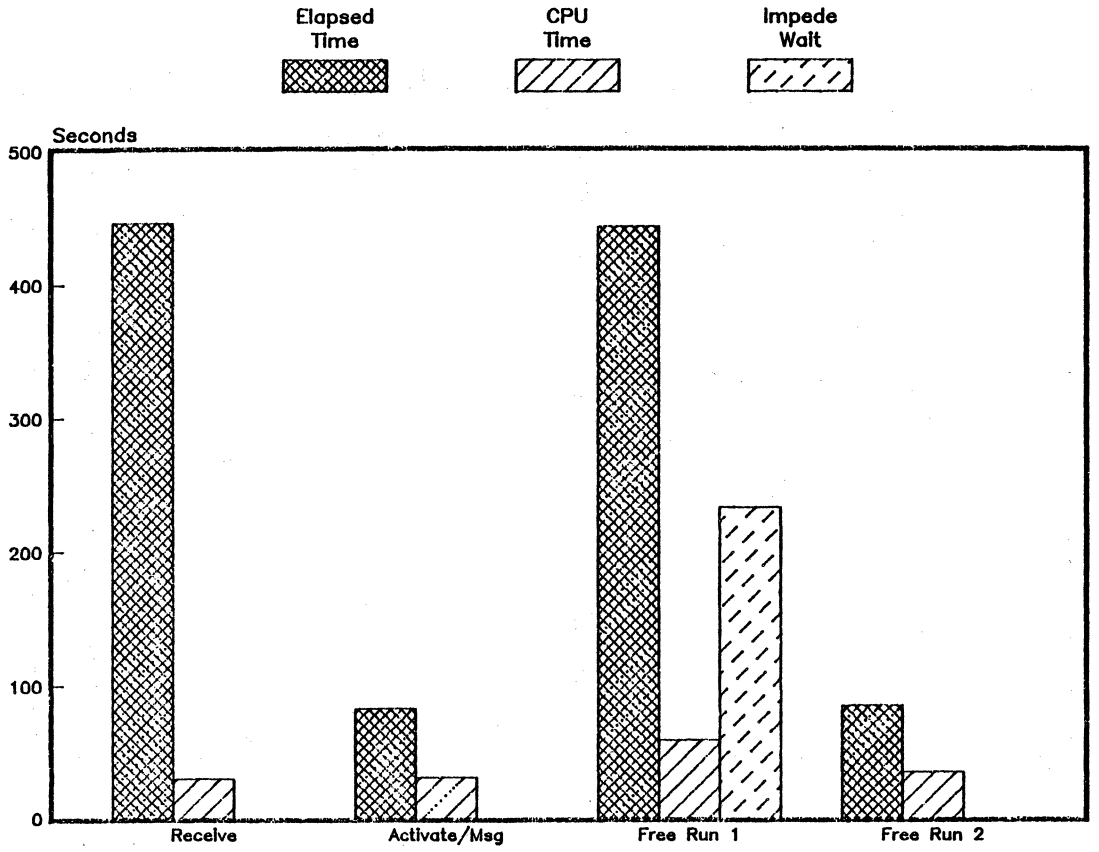


Test Type

Stand Alone Series 40 – C.00.20

Message Files vs MAIL

10,000 Receives/Reads



Stand Alone Series 40 – C.00.20

Additional Comments and Conclusions

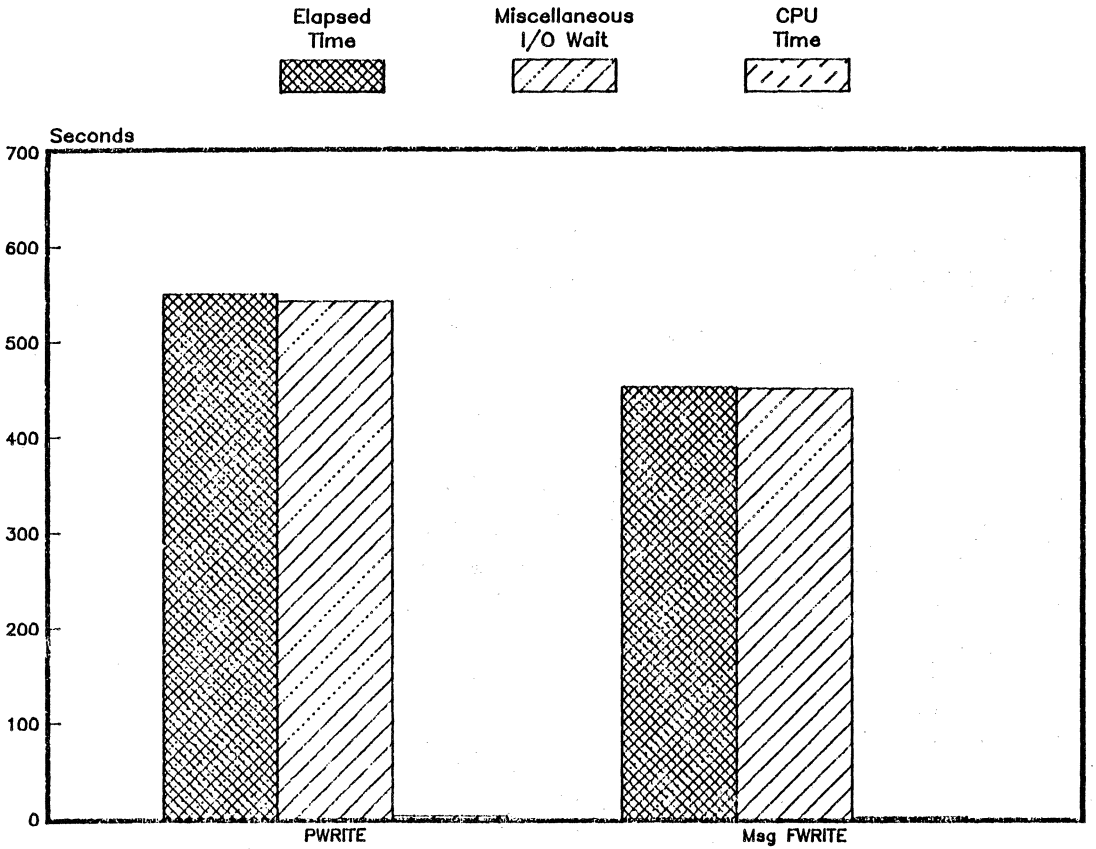
The send process was very slow, mainly as a result of the large amount of time spent in segment fault wait. The receive process was slow also, but in this case it is because it was father waited, a state MPEDCP does not explicitely record. Keeping the activates in, but replacing the sends and receives with reads and writes improved performance dramatically for both the reader and writer process. This is probably what many installations will see when they do a conversion of MAIL to Message Files. The free run tests show that performance can degrade severely if the reader process is unable to keep up with the writer. This happens, in general, when the lag between the two gets larger than the in-memory buffer. When this occurs the process uses up much time waiting to lock the file control block (shows up as impedes) and in another state not recorded by MPEDCP, Message File wait.

3. Message Files vs PTOP

To test the effects of Message Files on DS communication two additional SPL programs were written. The first was a PTOP master that did 500 PWRITE's to a slave, and the second was a PTOP slave that did 500 GET's and ACCEPT's. This gave a base line for PTOP activity. For the comparison to Message Files, the same reader and writer programs from the standard disc file tests were used. In this case the Message Files were assigned across a DS line. The DS line used for this test was a 4800 baud dial up line to another Series 40 of approximately the same configuration as the local. The results of these tests are shown on the next two pages.

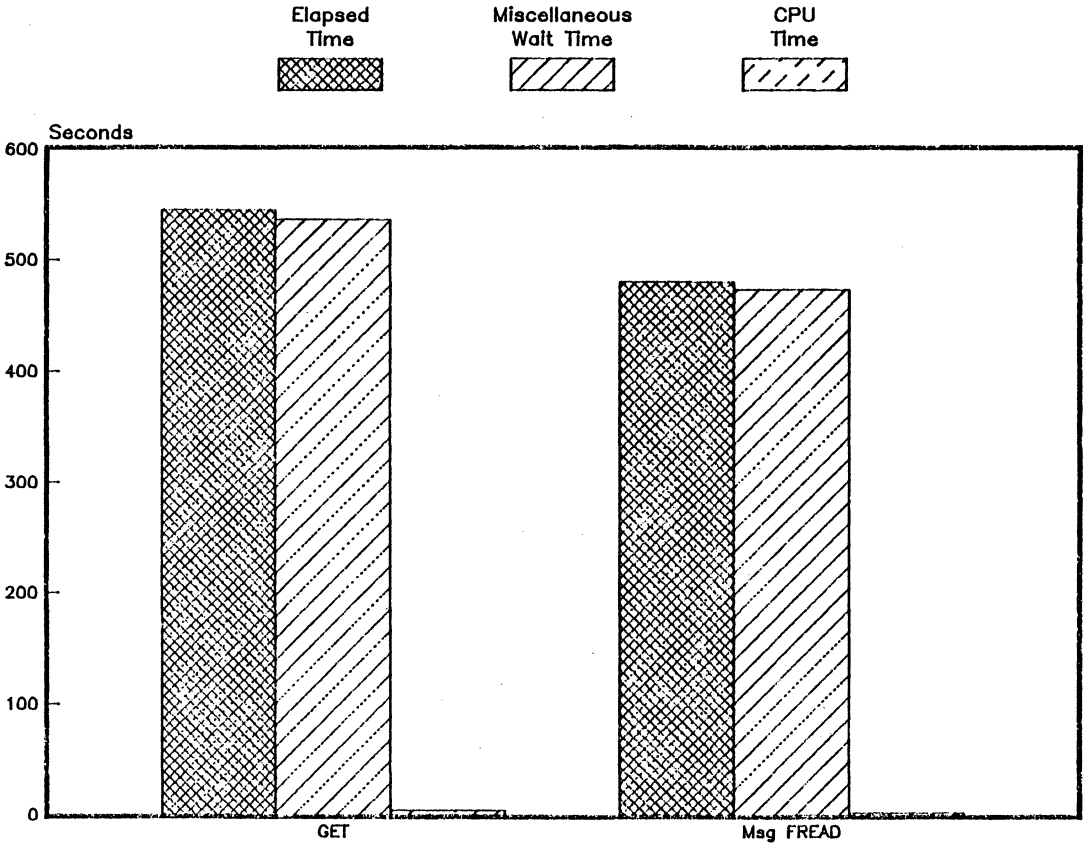
Message Files vs PTOP

500 PWRITE's/FWRITE's



Message Files vs PTOP

500 GET's/FREAD's



Test Type

Stand Alone Series 40 – C.00.20

Additional Comments and Conclusions

As expected, Message Files performed better than did the DS PTOP intrinsics. The differences, in this case, are not as pronounced as the difference between MAIL and Message Files. This is probably due to the type of line being used. A considerable portion of the miscellaneous wait time is line turn around time required by the 4800 baud dial up connection.

Summary

Message Files add important new capabilities to MPE, and provide performance improvement for existing applications with only minimal modification required. Even so, Message Files can be misused. The user should be careful to plan for the amount of "lag" required between the reader and writer processes. With this done, Message Files will give the best possible performance.

References

HP-3000 Communicator, Issue 26, Hewlett-Packard Company, 1980

HP-3000 Communicator, Issue 29, Hewlett-Packard Company, 1982

MPE Intrinsics Reference Manual, Hewlett-Packard Company, 1981

MPE File System Reference Manual, Hewlett-Packard Company, 1982

System Tables Reference Manual, Hewlett-Packard Company, 1982

TRANSACTION LOGGING TIPS

Robert M. Green
Robelle Consulting Ltd.

and

Dennis Heidner
Boeing Aerospace Company

Just Imagine

Time: 4:00 P.M. at your HP computer site.

Day: Heaviest sales day of the year, with hundreds of new orders entered into the computer by users at 20 widely dispersed CRT locations.

And then...

*** SYSTEM FAILURE ***

If the database is seriously damaged or the system won't restart without a RELOAD, what will you do?

a) Update your resume.

b) Call HP, then update your resume.

c) Recover, because you are using IMAGE logging.

Protect yourself from the protest of others when they find out that days' or even weeks' worth of data entry has been lost.

Use transaction logging, a feature of the IMAGE/3000 database system that allows you to transcribe all changes to your databases on a disc or tape file. This logfile can be

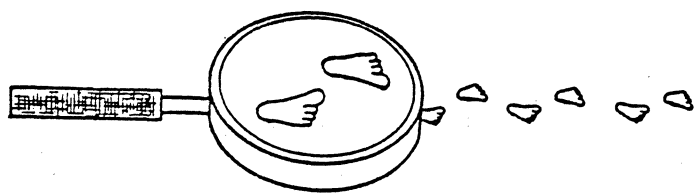
used later to re-create the transactions should a database be damaged due to a system crash, operator mistake, or program bug.

Three Misconceptions About IMAGE Logging

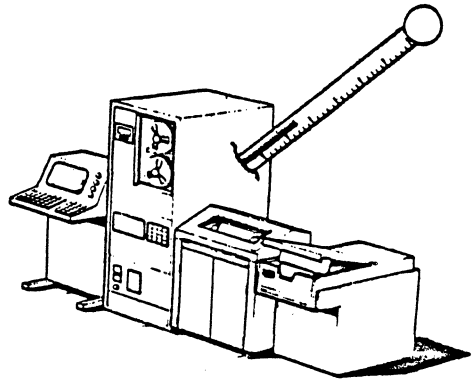
1. "Logging is a heavy load on the computer". Not so. Most users don't even notice it. Just turn logging on (using the simple checklist below) and see for yourself. Logging only slows down programs that modify the database; and the degree of degradation depends on how tight a modification loop you make ... if in doubt, try logging for a week and, if no one complains, leave it on. If the pattern for your programs is to ask the terminal user for information, then to interactively edit the various fields and, finally, to update the database, logging should have little impact on your system. Most problems that could arise with logging can usually be corrected by adding more memory to the computer.

2. "Logging requires program changes." Not so. You can start logging with no program changes. (Later, if you wish to take advantage of the full power of logging, you may modify your programs to define "logical" transactions.) You can try transaction logging without modifying any program. You do not have to call DBBEGIN and DBEND in order to log and recover; each physical transaction (e.g., DBPUT, DBDELETE, DBUPDATE) will be treated as a separate logical transaction by DBRECOV. The bracketing of several physical transactions by DBBEGIN and DBEND adds more logging power: you can recover a group of related physical transactions as a single logical transaction.

3. "Logging is only for recovery". Not so. Logging is useful for auditing, debugging, and tuning, too (if you have a tool like DBAUDIT or LOGLIST to print the logfiles). Because the logfile contains a complete record of every added, deleted, or modified entry in a database you can interrogate the logfile to answer difficult questions easily: Who is using QUERY to make unauthorized changes (and when and on which terminal device!)? Did program "X" actually put new order "Y" into the database, or does it have a bug, as claimed by the user? How many programs make modifications to dataset "Z", which has been found to contain inconsistent data?



By analyzing the transactions of the logfile, statistics can be gathered that help answer performance-tuning questions such as: Which dataset has the most activity? At what time of the day is transaction volume heaviest? About how long would it take to recover this logfile?



DBAUDIT is a fully-supported software product of Robelle Consulting Ltd. for generating audit reports from IMAGE logging. LOGLIST is an unsupported program with similar goals, written as an internal tool of the Boeing Aerospace Company.

When to Use IMAGE Logging?

You should be using IMAGE logging regularly if any of the following situations apply to you:

1. You have a high volume of transactions (i.e., it would take longer to recover the transactions by hand than to recover them with IMAGE logging).

2. You have no paper backup for your transactions (e.g., customer orders placed by telephone directly to CRT operators) and would find it impossible to re-create the transactions.

3. You have transactions with a high monetary value (e.g., changing the terms of a million-dollar commercial loan).

4. Your internal and/or external auditors require an audit trail for changes to key data fields of the organization.

5. You have many remote users entering and changing data (i.e., users that you cannot easily monitor or communicate with!).

6. You have a very large database (logging may reduce the number of times you have to do backup, and logging may be the only way to eliminate structural damage to your database if the time needed for a DBUNLOAD/DBLOAD is out of the question).

7. You have a very complex database (i.e., 3-10 paths into active detail datasets); complex databases are more prone to broken chains and other problems following a system failure.

Or, if none of the above applies, logging should still be used on an intermittent basis for debugging programs, quality assurance testing, and performance planning.

Checklist: Getting Started in Logging

You will need the cooperation of your system manager, your account manager, and someone at the master console. Arrange this cooperation first, or you will be very frustrated. In this example, it is assumed that you wish to turn logging on for a single database, generate some transactions to disc, then experiment with printing audit reports via DBAUDIT/LOGLIST.

1. Have the system manager give LG capability to your account, and your account manager give LG capability to your user name.

```
:HELLO MANAGER.SYS
:RUN LISTDIR2.PUB.SYS
>LISTACCT GREEN

  ....
 CAP: AM,AL,GL,ND,SF,IA,BA,PH,DS,MR
>EXIT
:ALTACCT GREEN;CAP=AM,AL,GL,ND,SF,IA,BA,PH,DS,
 MR,LG
:BYE

:HELLO MGR.GREEN
:ALTUSER MGR.GREEN;CAP=AM,AL,GL,ND,SF,IA,BA,PH,
 DS,MR,LG
:ALTUSER BOB.GREEN;CAP=AM,AL,GL,ND,SF,IA,BA,PH,
 DS,MR,LG
:BYE
```

2. Log on with LG capability, build a logfile on disc, acquire a log identifier with the GETLOG command and link it to the logfile. If you are already logged on, don't forget to log off and log back on (otherwise, you will not have the LG capability that you have given yourself above). Be careful to make the file big enough for all anticipated transactions and to allocate all 32 extents:

```
:BUILD TESTLOG;DISC=10000,32,32;CODE=LOG
:GETLOG AUDIT;LOG=TESTLOG;PASSWORD=BOB
```

3. Use DBUTIL to link your database to the log identifier.

```
:RUN DBUTIL.PUB.SYS
>SET TEST LOGID=AUDIT    (assume base=TEST)
PASSWORD? BOB
>EXIT
```

4. The first three steps are one-time operations; you do not have to do them again (for this combination of user, account, logfile, log identifier, and database). At a later time, you can link other databases to this log identifier. The steps that follow are repeated for each cycle of transactions that you wish to capture.

5. Ensure that no one is accessing the database, then use DBUTIL to enable LOGGING for the database. Once you have done that, no one can open the database until the log-identifier has been "activated" at the master console.

```
:RUN DBUTIL.PUB.SYS
>ENABLE TEST FOR LOGGING
>EXIT
:LISTLOG
       LOGID        CREATOR        LOGFILE
       AUDIT        BOB.GREEN      TESTLOG.PUB.GREEN
:SHOWLOGSTATUS
NO LOGGING PROCESS CURRENTLY RUNNING (CIWARN 1230)
:RUN APPLPROG.PUB.GREEN
**** UNABLE TO OPEN DATABASE: TEST ****
LOGGING ENABLED AND NO LOG PROCESS RUNNING
```

6. Now comes the hard part, unless you are located at the computer site. Convince someone to do a :LOG AUDIT, START command at the console. Perhaps a phone call to an influential friend would be useful at this point.

```
:LOG AUDIT,START
```

   You can tell if logging has actually been started with the :SHOWLOGSTATUS command.

7. Now log on a number of terminals and generate changes to the TEST database using QUERY, application programs, or ASK. All of these changes will be logged to the file TESTLOG.PUB.GREEN as they occur.

8. Once again, contact the console and have a command entered to stop the logfile.

```
:LOG AUDIT,STOP
```

   Terminate the programs that are generating transactions; when the last database accessor has closed the database, MPE will terminate the log process and close the log-file.

9. Run DBAUDIT and specify TESTLOG as the source of input records. If you do not want the report on the line-printer, you can use a :FILE command to the file DBREPORT to redirect it to $STDLIST or to a disc file.

```
:FILE DBREPORT=$STDLIST;REC=-79
:RUN DBAUDIT.PUB.ROBELLE
>INPUT TESTLOG
>EXIT
```

```
:RUN LOGLIST
>FILE=TESTLOG
>RUN
>EXIT
```

Questions and Answers About Logging

Q: How does IMAGE logging work?
A: After IMAGE has determined that a user call is error-free, but before it makes the actual database changes, IMAGE uses the WRITELOG intrinsic to send a description of the database changes to the logfile. If the transaction is large (over 238 bytes of data), WRITELOG breaks the transaction into a main record and continuation records. Logging itself is actually a feature of MPE (see User Logging in the MPE manuals), not IMAGE. User logging holds the transactions temporarily in an extra data segment and periodically flushes the transactions to a disc file. If logging to tape is desired, the disc file is copied to tape periodically.

Q: Do all IMAGE calls get logged?
A: No, only calls to DBOPEN, DBUPDATE, DBPUT, DBDELETE, DBCLOSE, DBBEGIN, DBMEMO, and DBEND. Calls to DBFIND, DBGET, DBLOCK and DBUNLOCK are not logged; programs that open the database in a read-only mode (5-8) are not linked to the logfile.

Q: Are QUERY modifications logged?
A: Yes. Also, QUERY-B, QUICK, and ASK/Cogelog allow you to define logical transactions via DBBEGIN and DBEND and to write "memos".

Q: Should I log to disc or tape?
A: There are arguments for both. On disc, the integrity of the logfile may be no better than that of your data. If you reach EOF on disc, logging stops, subsequent transactions are rejected, and data may be inconsistent. On tape, the tape drive is dedicated to logging for the entire day; transactions go to disc temporarily to smooth the data flow. If end-of-tape is reached, transactions stay on disc until the operator mounts another reel (the operator must be careful to mount a new tape, not just place the drive on-line; otherwise, MPE will write over the old logtape, destroying the contents!). If you have frequent parity errors on your tape drives, or problems reading tapes that have been written without error, then your logfile may be safer on disc (even if you have only one disc drive).

Q: Can multiple databases log to the same logfile?
A: Yes. However, only 128 user processes may share a logfile at one time (read-access users don't count, but users who open two databases with update access are counted twice). Note: It is possible to log the transactions but not be able to recover them later (especially if you allow logical transactions to extend over a long time period); see the error messages for DBRECOV in the manual.

Q: Must I stop logging before examining the transactions in the logfile with DBAUDIT or LOGLIST?
A: If the logfile is on disc, DBAUDIT/LOGLIST will read it, even if it is in use. If the logfile is on magnetic tape, cartridge tape, or serial disc, you must stop logging first.

Q: How does recovery, using the logfile, work?
A: IMAGE uses a roll-forward recovery method. The crashed database is purged, a backup database (stored before the transactions occurred) is restored from tape, and DBRECOV is used to reapply the logfile transactions to the database. If your applications defined logical transactions via DBBEGIN and DBEND, then incomplete transactions near the end of the logfile will be suppressed.

Q: Can I recover all transactions after a system failure?
A: No. The WARMSTART following a system failure attempts to "clean up" open logfiles. The clean-up phase of WARMSTART appends a "crash" record to the end of the transactions. For tape logging only, it copies any records left in the disc buffer-file to the end of the logtape. However, any records left in the memory buffers (extra data segment) are lost. A complete recovery cannot be guaranteed (of course, a systems programmer could look at the memory dump to see if the extra data segment were in memory and empty!).

Q: Does IMAGE back out logical transactions that are incomplete because a program aborted?
A: No. IMAGE treats aborting programs as having completed their current logical transaction. An abort logs a special DBEND, and DBRECOV does have a NOABORTS option, but the option has so many qualifications as to be of limited usefulness. DBAUDIT/LOGLIST does distinguish between normal DBENDs and abort DBENDs in its reports which can be very useful in monitoring program quality. At least one user has written an on-line back-out system, based on scanning a disc logfile backwards, looking for abort DBENDs.

Q: What is "Intrinsic Level Recovery" and how does it relate to transaction logging?

A: ILR (Intrinsic Level Recovery) is an enhancement to IMAGE which is currently in the works. When ILR is enabled, IMAGE creates a "zero" dataset (name = base00) and uses it to record before pictures of each disc block that is changed during a single call to DBPUT or DBDELETE. Should the system fail, IMAGE uses the ILR file to rollback the database to the state it was in before the last intrinsic call began. This enhancement should eliminate the problem of broken chains after a system failure. It does nothing for the logical consistency of the database (that would be full transaction backout), but it should be possible to combine transaction logging, ILR, and DBAUDIT/LOGLIST to unwind the last logical transaction by hand.

Q: When I plan to run DBRECOV, how do I know that I have restored the proper backup database?

A: For DBRECOV to succeed, you must restore a backup database created just prior to the start of the logfile transactions (i.e., no unlogged transactions between backup time and LOG START time). If you always use DBSTORE and DBRESTOR, IMAGE guarantees that you are using the correct database with the correct logfile. If you use SYSDUMP (or :STORE) for backup, it is up to you (or your operator) to guarantee that the database matches the logfile (good luck!).

Q: After a DBRECOV, how can I find out which transactions were not recovered (i.e., were incomplete)?

A: Use DBRECOV's FILE command to write unrecovered transactions to recovery files, then use DBAUDIT/LOGLIST to examine them. If you are not using DBBEGIN and DBEND, there will be no unrecovered transactions.

Q: How can I tell quickly whether all of a process's transactions were recovered by DBRECOV?

A: An asterisk ("*") on the far right side of the process's statistics indicates either that no DBCLOSE was found, or that all of the transactions could not be recovered. If the asterisk is missing, all of that process's transactions were recovered.

Q: Can I suppress all of the transactions of a particular process during recovery?

A: No, due to interaction between transactions by different processes.

Q: Can non-IMAGE applications log to the same logfile used by IMAGE for my database transactions?

A: Yes, if the logid and password are known (use OPENLOG, WRITELOG). DBRECOV will just ignore these non-IMAGE transactions when it does recovery (i.e., recovery is up to you!).

Q: Can my applications declare logical transactions that span two or more databases?
A: Not officially; IMAGE treats each DBBEGIN and DBEND pair as linked to a particular database. If you use two DBBEGINs, then update two databases, and complete the transaction with two DBENDs, there is still a small chance of a system failure during the window between the two calls to DBEND, resulting in recovery of the transaction for one database, and suppression for the other. If you use this strategy, you should have DBAUDIT/LOGLIST to examine the recovery files in order to undo some transactions by hand, if necessary. Another possibility is to use the new BEGINLOG and ENDLOG intrinsics to define multi-database transactions, but they are not recognized by DBRECOV. You can determine where the last ENDLOG record is in the logfile with LOGLIST. Then use the record option in DBRECOV to recover up to a specific record number.

Q: Can I start and stop logging from a job or session, or only from the console?
A: Normally, the console operator starts and stops logging from the master console, using the :LOG command. With the :ALLOW command, the operator may "allow" other users the :LOG command (but error messages are still printed on the master console). Logging need only be started (or stopped) when the system is shut down or the database is going to be backed up. One user left his log process going for over 40 days!

Q: What happens if a disc logfile runs out of space?
A: Whoops! Just like a system failure: logging stops, programs return from IMAGE with error -111, database may be inconsistent (but physically sound), and you should get the users off and do a recovery from the logfile. You will lose the transactions in the memory buffers, but you may still want to dump the logfile with DBAUDIT to see what processes were active. Moral: do not let your disc logfile overflow!

Q: What happens when a tape logfile reaches the end of tape?
A: A message will be issued to the system console to mount another tape. If this message is ignored for several minutes, the processes that are being logged will suspend. A message is not issued to each individual terminal user, nor is the tape request repeated on the console if you ignore it (a RECALL is suggested).

Q: What happens if there is a tape parity error?
A: The result is the same as reaching end of file on a disc logfile. So, use good tapes and run TAPETEST on them.

Q: When the system fails, can I safely restart logging and database modifications without doing a recovery first?
A: No, unless the database was not open for updates. There are three reasons to do a recovery: 1) to suppress incomplete logical transactions, 2) to insure against structural damage, and 3) to ensure that transactions logged after the failure will be recoverable should the system fail again catastrophically (recovery depends upon having every transaction in a sequence logged; after a system failure, a few transactions may be lost in the memory buffers).

Logging Tips for Operations

1. Gather into one binder copies of all documentation relevant to transaction logging. Find the two articles on logging in the San Antonio proceedings. Carefully read all of the information on transaction logging before using it in production:

   Console Operators Manual - :LOG command
   MPE Intrinsics Manual - the chapter on LOGGING
   MPE Commands Manual - GETLOG, RELLOG, ALTLOG, etc.
   System Supervisor Manual - ALTACCT/ALTUSER (CAP=LG)
   IMAGE Database Manual - the section on LOGGING
   - the section on DBRECOV and DBUTIL
   - the material on DBBEGIN and DBEND
   The Communicator, Issues 20-28 - new features
   Software Status Bulletins (SSB) - for bugs in logging

2. Regular practice and careful procedures are important. Prepare a written recovery checklist for the operator; then fully backup your system and the databases (using DBSTORE), and have the operators do a practice recovery. Also, run DBCHECK at least once a quarter to check for structural damage to the database, and run DBLOADNG or HowMessy/Robelle at least once per month to check for inefficiency in the databases.

3. Since logging runs as a system process, you cannot use :FILE equations to redirect the logfile.

4. If you will ever use streams to manage logging, you must assign a password to the logid (due to a bug in DBUTIL, see Software Status Bulletin):

   :ALTLOG logid; PASSWORD = pass

5. Prepare for the fact that logging may cause occasional pseudo-crashes (e.g., reaching EOF on disc). For example, one user had a problem with his tape drive.

The drive was no longer writing consistently to the log. And, the tape drive did not catch its own mistakes. Later, when the database had been purged and log tape was needed for DBRECOV, 3000 transactions were completely lost because the tape was unreadable.

If you have a problem like this with logging, you may cause more damage to the database if you try to use DBRECOV than if you ignore the problem. You can prevent this by using DBAUDIT, before any recovery, to verify the tape. The current "crashed" database may be the best one you have. If you find that the log tape (file) is corrupt, don't recover. Instead, correct any logical inconsistencies in the current database by hand (using DBAUDIT/LOGLIST to examine the logfile for guidance), and, if DBCHECK reports any structural damage use DBUNLOAD and DBLOAD (or Adager and SUPRTOOL) to repair the damage.

6. When you change the structure of your database, keep a copy of the old schema. To analyze a logfile, you must have a database whose structure matches that of the original database (IMAGE logs by set and item number, not name). The contributed program NEWDBGEN (in the San Antonio swap tape) takes an Adager-generated schema, changes the base name, and reduces the capacities of all datasets to 3.

   Caution: Adager disables logging and removes the logid from the database (in fact, this is the only way to remove the logid!). Be certain to enable logging and reset the logid after using Adager.

   After a DBUTIL ERASE function, most changes by Adager, or a DBLOAD (even without any structural changes to your database), you must do a DBSTORE before restarting logging. This is because IMAGE logs detail deletes and updates with relative record numbers.

7. Logging cycles

   A logging cycle is the time between backups that are covered by the transactions in a logfile. How often you start a new cycle depends on how long it will take to process the logfile, should you need to recover. For low-volume applications, start a new log cycle whenever you do a full backup (perhaps once a week). For high-volume applications, start a new log cycle (DBSTORE and LOG START) whenever you do a partial backup. DBAUDIT/LOGLIST generates statistics to estimate how long the recovery of a specific logfile is likely to take.

   The HP manual recommends disabling a database for access

when it is DBSTOREd, but we disagree. If the operator fails to enable the database again for access after the DBSTORE, no jobs or sessions will be able to open the database until someone can enable it for access. If you should ever disable a database for logging (not particularly recommended), be careful to do a new DBSTORE before STARTing logging again.

To recover a database, check the logfile with DBAUDIT/LOGLIST, set the JOBFENCE down, rename the existing database to another group (using BASENAME of Adager or the contributed DBRENAME), or, if you do not have sufficient disc space, DBSTORE it to tape and purge it with DBUTIL (this is in case recovery fails!), DBRESTOR the backup base from tape, use DBUTIL to disable the base for access and enable it for recovery, then run DBRECOV. When done, use DBUTIL to enable the base for access and disable it for recovery, then start a new logging cycle if possible (DBSTORE and start new logfile). If you don't have time for a final DBSTORE, HP suggests that you can RESTART the current logfile (see the manual for details of this approach).

8. If you have some batch jobs that you do not want slowed by logging, DBSTORE the databases, disable logging, run the BATCH jobs, and, if all is well when they are done, DBSTORE the base and resume logging. If you have a crash during the jobs, just backup to the previous DBSTORE and rerun them. Of course, if you expect to do a full audit of all database changes from your logfiles, you must not turn logging off to run large batch jobs.

9. STOP, START, and RESTART with the LOG command

   :LOG STOP will stop the log process but not always immediately. Logging does not stop until the last user with update access closes the database. If one user fails to log off, you cannot do a :LOG START, nor can any new users open the database (the STOP may be pending for hours). Therefore, use :SHOWLOGSTATUS after :LOG STOP to verify that logging has actually stopped, and take action (:WARN?) if it hasn't.

   When logging to disc, START is illegal if the file is not empty. RESTART is okay after a STOP (because it causes new transactions to be appended to the logfile), so long as you have not made any unlogged changes to the database by disabling logging. RESTART is not okay after a system failure because some transactions may be lost, making recovery via the RESTARTed logfile questionable. To maintain a complete audit, STOP logging, :STORE and :PURGE the logfile (or :RENAME it), :BUILD a new logfile, DBSTORE the database, and issue a LOG START.

When logging to tape, START is accepted even if the file has log records in it already, but the existing transactions are erased (because, although logtapes are labelled, they do not have an expiration date). RESTART is okay with tape (so long as no unlogged transactions are missed), because logging reads to the end of the tape and begins appending. To maintain a complete audit trail, STOP logging, do a DBSTORE, and START logging again with a new tape (tested and cleared of any old data with the contributed TAPETEST program).

Logging Tips for Programming

1. DBINFO mode 401 provides information about your current logging status: logid, enabled/disabled, user logging flag, transaction-in-progress flag, and current transaction number; but not the amount of space left in the logfile.

2. If a logical transaction contains only one physical transaction (e.g., only a single DBPUT call) and you have no memos to log, you may omit the DBBEGIN and DBEND. DBRECOV treats "stand-alone" IMAGE calls as complete logical transactions.

3. Always check the status returned by IMAGE after DBOPEN, DBPUT, DBDELETE, DBUPDATE, DBBEGIN, DBEND, DBMEMO, and DBCLOSE. IMAGE returns an error -110 for an OPENLOG failure (DBOPEN only), -111 for a WRITELOG failure, or -112 for a CLOSELOG failure (DBCLOSE only). In all three cases, the second word of the status array gives more information: 3=log process not running, 8=bad password for logid in the database, 9=write error (such as parity error), 13=too many users, 15=end-of-file on disc logfile. Write your programs so that if IMAGE returns an indication of logging errors, you can write out an independent error-file and abort the process. If you don't, you may have a hard time finding out who, what, and how bad your database crash was.

4. Use WRITELOG to record changes to MPE files (not DBMEMO); get the index for the logid from OPENLOG and be certain to pass it unchanged to WRITELOG (an invalid value causes a system failure). Use DBBEGIN to record CRT input for the transaction (including values for critical fields that are not logged when you update a detail entry); use DBEND for final transaction status (error messages, etc.); and use DBMEMO for remarks by the CRT user. It is also useful to log negative information. For example, if a user cancels a new order after entering part of it, log a DBMEMO telling which order number was cancelled; this can be helpful in resolving later disputes with users.

5. The DBBEGIN intrinsic does not turn transaction logging on! If the database is enabled for logging, all transactions are logged, even if they are not bracketed by a DBBEGIN and DBEND. According to the IMAGE manual, IMAGE returns an error 71 to your program if you attempt to call DBBEGIN, DBEND, or DBMEMO after opening the database with read-only access, and IMAGE returns -111 if a WRITELOG error occurs (see point 3 above).

6. If the item being modified by DBUPDATE is a compound item (e.g., 4x10), the whole item is logged. Any change made to an item defined with a 'sub-itemcount' greater than one will cause all elements of that item to be logged.

7. Do not use FLUSHLOG (to flush memory log buffers to disc) unless you have a compelling reason; it will increase the overhead of logging dramatically.

8. Use BEGINLOG and ENDLOG to bracket super-transactions that cover multiple databases. (There was a bug in MPE such that both BEGINLOG and ENDLOG generated the same logging code, but it appears to be fixed in D-delta). To log multiple databases, use the following:

```
Call OPENLOG
Call BEGINLOG (dindex...)(once)
   Call DBPUT (base1...)...
   Call DBDELETE (base2...)...
Call ENDLOG (dindex...)
instead of:
Call DBBEGIN (base1...)
  Call DBBEGIN (base2,...)
     Call DBPUT (base1...)...
     Call DBDELETE (base2...)...
  Call DBEND (base2...)
Call DBEND (base1...)
```

9. Add a TIMESTAMP field to each dataset and update this field whenever you do an update to the record. Because DBUPDATE logs both the 'before' and 'after' values of the TIMESTAMP, you will have a traceback (by date and time) from each change to the previous one. If you retain your logfiles, you can go back and check all changes to a specific IMAGE entry (i.e., the history of an inventory part). Another idea is to include room in your TIMESTAMP field for the identifying values of the entry, if it is a detail entry. This helps get around the problem of IMAGE's not logging critical field values on detail updates (because they are never modified).

Logging to Disc

If you log to disc, build your logfile large enough to hold

all transactions between DBSTOREs! If the logfile fills up, it causes the equivalent of a database crash. Build your disc logfile with a capacity larger than your needs, with 32 extents, and all extents allocated:

```
:BUILD LOGFILE;DISC=100000,32,32;CODE=LOG
```

You can calculate the required size (in records) for your logfile by following this formula:

```
# of records =
   4 * number of database opens
     + (number of updates * update rec len)
     + (number of puts * put rec len)
     + (number of deletes * delete rec len)
     + (number of dbbegins)
     + (number of dbends)
     + (number of dbmemos * memo len)
```

The update rec len (in sectors) = (# of items in list + 15 + 2*(data buffer wordlen))/119

The delete rec len (in sectors) = (13 + data buffer wordlen) /119

The put rec len (in sectors) = (# of items in list + 14 + data buffer wordlen)/119

The memo len (in sectors) = (message wordlen + 7)/119

All of these lengths are rounded up to the next sector. If the buffer sizes are not known, use the entry length (see FORM SETS in QUERY). You can count the number of items in the field-list, or, if the list was "@", use the item count for that particular set. To check whether you have calculated properly, use :SHOWLOGSTATUS to monitor the number of records in the logfile. If you have especially large batch runs on weekends, you can allocate small logfiles during the week and a larger logfile on Friday evening.

In order to maintain a good audit trail, you should keep at least 5 to 10 logfiles back on :STORE tapes. But remember that an active logfile cannot be backed up. Keeping an audit trail of logfiles also allows you to use several logfiles in sequence to recover from an older backup database, should you find the most recent backup database unusable (due to tape error or inexplicable database damage not noticed before the damaged database has been backed up).

Logging to Magnetic Tape

Transaction logging to tape uses ANSI-labelled tapes; a tape lockword is not allowed and the expiration date is always "00/00/00" (meaning you can write over the tape by accident). When you activate logging with LOG START, you must often mount a brand-new tape and cajole MPE into writing the original volid onto it:

```
{mount brand-new tape on drive}

 I/O ERROR IGNORED DURING AVR
 Volume (unlabelled) mounted on LDEV#7

 :GET LOG logid;LOG=filename,TAPE
 :LOG logid,START

 Volume ID for filename (max chars=6)?
 =REPLY pin,LOGTAP        {you enter volume id!}
 Mount tape of volumeset LOGTAP (ANS)  {looks like mes-
                                        sage!}
 =REPLY pin,7  {MPE writes volid into new label on tape}
```

When logging reaches the end of a logtape, it makes a request of the operator to mount another reel. If the operator ignores the request, the user processes that require logging will suspend within a few minutes. No message is printed on the user terminal. If the user telephones the operator and a new reel is mounted, the processes will resume execution with no problem (i.e., there is no database crash).

Warning: acceptance of a new reel by logging requires only that you put the tape drive on-line; there is no reply. Do not put the tape drive on-line without first checking to be certain that a new logtape has been mounted. Otherwise, logging will write over the old tape, wiping out your logged transactions!

After a system failure, you can use a WARMSTART to recover the contents of the disc buffer file and append them to the active logtape. However, there are a few things to watch out for:

> If you don't use TAPETEST to erase old logtapes after use, they will still contain old transactions. WARMSTART attempts to read through the tape until it finds the end. If there are old transactions out there after the new ones, you may be unlucky enough to have WARMSTART read past the new to the old with a parity error or invalid inter-record gap (this happens about half of the time). Because WARMSTART only checks transactions for numerical sequence and not for chronological order, it may append the disc buffer transactions to the end of the tape, after the old transactions, which may be from last month!

Do not just put the tape drive on-line. Be certain to force an AVR (reset, rewind, then on-line). Otherwise, WARMSTART may wipe out the first record on your logtape, converting it into an unlabelled tape (see Logging Error Messages below for a way to get out of this mess).

There once was a bug in Power Fail Recovery when logging to tape, but the latest HP manuals have redefined this bug as a feature. The logtape cannot always be recovered completely after a Power Fail Recovery. That is, PFR has to be treated like a system failure.

Here is an alternate procedure for dealing with a PFR. Put the logtape back on-line; watch while logging writes the waiting transactions to the tape; issue a LOG STOP and get all users to close the database. Check the logtape with DBAUDIT/LOGLIST, if no errors are detected and you get to the trailer record, RESTART logging and let the users back on. If DBAUDIT/LOGLIST finds errors in the logtape, backout the transactions by hand (using the DBAUDIT/LOGLIST report) and DBSTORE before letting the users on, or do a full recovery.

Thirty thousand records (30,000) use up about 1/3 of a 1200-foot tape (1600 bpi). DBAUDIT/LOGLIST prints an estimated time to recover when it audits a logfile. The time required to recover depends on your application. One tape with thirty thousand transactions took about three hours to recover (with a database of about thirty megabytes), but about 90% of the transactions were DBDELETEs and DBPUTs to detail sets with 10 paths. The other 10% consisted mainly of updates. If most of your transactions are updates only, the recovery runs much faster.

Dedicate a number of tapes solely for logging (10-20), then rotate the ones you use. This gives you an audit trail of your database. These backup logfiles give you another important capability: when you go to do a DBRECOV and you find that the last DBSTORE tape is not readable (gulp!), you can now go back to an earlier DBSTORE tape and roll forward with two (or more) logfiles.

Logging Error Messages

Below are the possible logging messages that may appear on the operator's console, with some comments on them:

UNABLE TO OPEN USER LOGFILE ! (ULOGERR 1)
This error can only occur on a LOG START or LOG RESTART for tape logging. It means that the tape drive was not available (possible cause: operator did a REPLY 0).

UNABLE TO OPEN USER LOGGING BUFFER FILE(ULOGERR 2)
This may be a configuration problem or lack of file

space. Check your free space, the system configuration, then contact your HP SE. This happens only with tape logging.

CAN'T OPEN USER LOGGING FILE ! (ULOGERR 3)
The logfile may not exist or it may be in use by another process. Check your typing or first build the file. Check to make sure you have the correct file-security access to the logfile in question.

NO DATA SEGMENT AVAILABLE FOR USER LOGGING PROCESS ! (ULOGERR 4)
You have a system configuration problem, or a very bad problem in program design. Check your system configuration tables and/or contact your HP SE, and memory salesman. You may be out of DST entries or out of Virtual Memory (run TUNER4). This error indicates a general problem with your system.

FILE LABEL ERROR ON USER LOGFILE ! (ULOGERR 5)
The MPE file system, or the logging process, has inadvertently destroyed the ANSI label on this tape. (This error should only occur on releases of MPE prior to D-delta). Don't panic - the steps to recover this tape are:

a. Build a file on disc large enough to hold all the records on the tape.
b. Copy the tape to the file using FCOPY and NOUSERLABELS.
EXAMPLE:
FROM=*TAPE;TO=CRASHLOG;NOUSERLABELS
c. Release the logging process and change it to the disc file you just built. EXAMPLE:
:RELLOG STORELOG
:GETLOG STORELOG;LOG=CRASHLOG,DISC
d. Run the recovery procedure.
e. Switch logging back to tape!

ERROR WHILE WRITING TO USER LOGGING BUFFER. (ULOGERR 6)
This indicates a disc hardware problem or a bug in MPE (the file label or directory entry may be destroyed). Check your system configuration, then call the HP SE. You may have to perform a database recovery procedure for this one, unless it occurred on the very first log record. Strangely, this message can occur for either a write error or a read error on the buffer file. You should also be concerned about possible system failures, since you probably have disc hardware problems or MPE bugs.

ERROR WHILE WRITING TO USER LOGGING FILE ! (ULOGERR 7)
This error occurs with tape logging only. As with the previous message, this one can be caused by either a

write or read error. You have encountered either a parity error on the tape or a hardware failure on the tape drive. Determine which and correct the problem. You will also have to get all users off the database and do a recovery. Important: before you purge your database with DBUTIL, make certain that the logfile is intact (run it through DBAUDIT). User programs that attempt to log receive an error -111 or -110 from IMAGE.

OUT OF USER FILE SPACE FOR USERLOGGING PROCESS ! (ULOGERR 8)
You have hit the end-of-file on a disc logfile. You should have built a larger logfile in the first place. You will probably have to do a recovery. After recovering from this "crash", build a larger logfile before starting logging again.

OUT OF DISC SPACE. (ULOGMSG 9)
Logging has been unable to allocate the next extent for a disc logfile. Allocate additional space, or review the logging requirements. This error will probably require that the user run a database recovery procedure. If you had allocated all 32 extents when you built the logfile, this could not have happened.

USER LOGGING PROCESS ! IS RUNNING.(ULOGMSG 10)
This is not an error; it is a welcomed message (if you just did a :LOG logid,START command).

USER LOGGING PROCESS ! IS TERMINATED. (ULOGMSG 11)
This is the normal response to a :LOG logid,STOP command.

RECOVERED ! RECORDS FOR LOGGING PROCESS ! %
INCLUDING ! OPENS AND ! CLOSES (ULOGMSG 12)
This is the message that you should see after:

a. encountering a system crash or hardware failure;
b. warmstarting the system (so you could recover);
c. making the necessary reply to transaction logging (from the operator's console; if you did not REPLY, you may be setting yourself up for another crash!).

ERROR WHILE RECOVERING USER LOGFILE !. (ULOGMSG 13)
This message used to occur during WARMSTART, but is no longer generated under any circumstances in the D-delta release of MPE.

USER LOGFILE ! NOT RECOVERED. (ULOGMSG 14)
Do not panic, but you may have lost a bunch of log records! You should only see this error when recovering an open logfile during a WARMSTART after a system failure. There is no known corrective action; and the causes are not well-known so there is no preventive action. You have lost the transactions in the intermediate disc file (when logging to disc).

If you are logging to disc, we don't know what this means. Let your HP SE know you had a problem, and do not attempt DBRECOV with this logfile without checking it first with DBAUDIT.

RECOVERING USER LOGFILE ! (ULOGMSG 15)
This is the response you want to see following a crash and a WARMSTART (to recover).

USERLOG ! RESTARTED. (ULOGMSG 16)
This okay, it is just a message. Remember, though, that you should not have modified the database while the logging process was inactive. If you did, and you have a crash, you may have a big mess on your hands.

LOGGING FILE FOR LOGGING PROCESS ! IS EMPTY. (ULOGMSG 17)
You are trying to recover from a file with nothing in it. Do not waste your time. If there should have been something here, you might want to call HP.

LOGGING PROCESS ! SUSPENDED, TERMINATION PENDING. (ULOGMSG 18))
This message is okay; the logging process was suspended for some reason (waiting for another tape.). When it resumes, logging should terminate normally.

INVALID DISC LOGGING FILE FOR LOGID !.(ULOGERR 19)
This is not a valid "LOG" file (code is not 1090, blocksize or record size is not 128, file is not ASCII, or file is too small). Recheck your steps in setting up a logging process.

UNABLE TO START/RESTART LOGGING PROCESS !.(ULOGMSG 20)
Because of errors (previously printed), the logging process cannot be started or restarted. Take corrective action to fix the earlier errors.

LOGGING PROCESS ! IN USE, TERMINATION PENDING. (ULOGMSG 21)
The operator has requested that logging stop. However, there is still a user process that is using this log-file. As soon as these processes log off, the logging process will terminate.

USER LOGGING FILE ! NOT EMPTY (ULOGMSG 22)
You asked the logging process to start logging on a disc file which already has some data in it. Recheck and rethink.

***** WARNING CHEKSUM FAILURE ON LOGFILE ! ***** &
**** NO END OF FILE ENCOUNTERED **** (ULOGMSG 23)
We think this error occurs during WARMSTART recovery. The logfile may have been altered, destroyed, or tampered with. Or, the logfile may be okay. Check the logfile with DBAUDIT/LOGLIST before attempting a recovery.

About the Authors

Robert Green is president of Robelle Consulting Ltd., a five-year-old software firm located near Vancouver, B.C.; he has over 10 years' experience on the HP 3000 starting in the HP factory and has authored numerous papers about the 3000. Bob is the developer of the software products QEDIT, SUPRTOOL and DBAUDIT.

Dennis Heidner is an employee of the Boeing Aerospace Company in the Seattle area where he has technical responsibility for an HP 3000 that manages electronic test equipment. In order to track the movement of valuable test instruments, Dennis wrote a program called LOGLIST which reads IMAGE logging files.

QMIT: THE MPE QUALITY MIT

Roy A. Clifton
Engineering Section Manager
Hewlett Packard Computer Systems

INTRODUCTION

Criteria for Quality

In April, 1982, Hewlett Packard Computer Systems Division Management made a decision to produce a greatly improved Quality Master Installation Tape (Q-MIT). The following guidelines were established:

A) Produce a clean release of MPE IV on which to integrate any new products.

B) Develop a procedure for testing and certifying major software releases (MITs, ITs, etc.) that does not depend on field software coordinators to finish the integration and testing process.

C) Develop a procedure for releasing product tapes that includes assurance that they will work with the current, supported release(s) of MPE and other supported products (Software compatibility matrix).

D) Develop a procedure for resolving service requests and checking corrections that is timely, and does not unseat the system in the process of installation.

The clean version of MPE (Q-MIT) was accomplished in part through the clearing of the service request backlog and by fixing existing known problems. The result is a solid software base for future products.

The introduction of the Q-MIT, Quality-Master Installation Tape, represents a very important event within the larger framework of software quality. For some time now, the various divisions within the HP Business Computer Group have been engaged in a quality improvement campaign. This presentation will highlight those quality improvement efforts in the area of software that will continue to increase the perceived quality of HP's software offering.

Recognizing the importance of the field's perception toward our software, this presentation will focus on the quality improvement activities you felt were of major importance. Several SE's in the local area were asked for a set of software quality criteria. Most of the criteria that came back with those responses are being addressed at this point, others are well on their way to being addressed. It is our goal here at CSY to keep expanding our efforts and thus address more of your concerns in the future. Among the more common of your concerns are:

- SRs responded to in a timely fashion
- Consistency - all factory organizations have synchronized efforts
- Make sure any quality issue resolved does not resurface
- Make sure documentation is complete, accurate and tested
- Published standards for programming and documentation to aid in maintenance and readability
- Rigorous QA procedures
    - improving our test tools constantly
    - complete integration testing and testing of all critical functions
    - testing of patches and fixes before they are integrated
- A consistent IT and Delta strategy which the SEO and the customer can depend on for regular updates

Areas where software quality is affected

To address those concerns with quality, this presentation will break down the various areas where software quality can be affected. Within those areas, we will give a brief discussion on the quality activities and expected results from those activities.

The areas where quality can be affected are presented as follows:

A. SOFTWARE DEVELOPMENT CYCLE (R&D)
B. SOFTWARE INTEGRATION, TESTING AND DISTRIBUTION
C. SOFTWARE DOCUMENTATION
D. QUALITY CONSCIOUSNESS OF EACH ORGANIZATION

A. SOFTWARE DEVELOPMENT CYCLE

Quality in software, of course, must stem from all the various areas associated with the software product. Software quality has its roots, however, in the labs where it is developed and documented. The various software labs which produce software for the HP 3000 have taken major steps to safeguard the quality in our product offerings.

DEVELOPMENT STANDARDS

To improve quality in Hewlett-Packard software, the lab is following certain standards in the development of its software. By following these development standards, the software produced by the software lab will be less prone to errors and easier to maintain, and thus of higher quality.

MPE Module Ownership

Module ownership is an important step in properly maintaining MPE. Each MPE module is now owned by an individual in an MPE group (e.g. Data Access, Kernel, User Interface, etc.). All fixes to individual modules must be inspected by the individual owning the module. In most cases, the owner will be the one to make changes to the module. Should another engineer make changes to his/her module, then the owner will inspect the changes to insure their quality before they are integrated. This method of module ownership eliminates conflicting fixes which could be introduced into the code when several engineers make changes to a single module.

Software Product Life Cycle (SPLC)

The Software Product Life Cycle (SPLC) discusses the five major steps of a software project; Investigation, Design, Implementation, Testing and Release. By following the SPLC, the engineering team produces a number of useful documents that will accompany the new code. The first important document produced by following the SPLC is the External Reference Specification (ERS). The ERS describes the function and use of the product. The second document produced, the Internal Maintenance Specification (IMS), is extremely important for maintaining the code. It describes in detail the algorithms and data structures used to implement the product. Through these documents, the SPLC promotes higher quality code. The SPLC stresses the use of technical team code reviews and code walk-throughs. This procedure encourages criticism and promotes the concept of "egoless" programming.

Structured Programming Standards

Members of the various software development teams have produced a set of "Structured Programming Standards." These standards discuss structured programming (e.g. flag programming, procedure structure, block structure, etc.). They also govern the use of indentation, variable declarations, naming of variables and good comments. Lastly, they describe the proper use of special SPL and PASCAL constructs. As this document is followed, the software will become easier to read and easier to maintain. All future development will follow these programming standards.

Prototyping

Useability has become an increasingly important measure of quality as our software products move from the DP department into the office, where our products are targeted for the non-technical user. The software development process in the Information Networks Division's (IND's) Office System's Lab has shifted accordingly.

In the past, the externals of our products were designed on paper and documented by an External Specification (ES). These products were coded according to the ES and tested at ALPHA test sites (HP internal users). During the ALPHA test, significant changes to the user interface were made which invariably resulted in slipped schedules and less than ideal user interfaces.

IND's response has been to prototype all new products whose target user is non-technical. The prototyping is done prior to completion of the External Specification. The prototyping itself consists of writing a minimally functional version of the product, giving it to a set of unschooled users, recording their reactions, and tuning the user interface accordingly. Some IND products that used prototyping were HPWORD, IDSCHAR, HPDRAW and HPEASYCHART. Many products currently under development are using prototyping.

Quality Plans

The purpose of the Quality Plan is to set quality goals for a product and measure the product against those goals. The Quality Plan is generated during the external design phase by the development team. This document identifies the quality objectives of the product, describes the means of achieving them, and the method of verification. The basis for evaluation is fitness for use in five areas. These areas are functionality, useability, reliability, performance, and supportability. This is accomplished by having defined objectives to be met for each product development phase, a plan for achieving the objectives, and a method of verification against an identifiable set of criteria before allowing the product to move to the next phase.

CURRENT PRODUCT ENGINEERING (CPE)

CPE is the term used to describe HP software maintenance. CPE time is spent working on customer problems (e.g. hot-sites) and Service Requests (SR's).

Q-MIT

The Q-MIT was a direct result of the current quality efforts. This concentrated effort to reduce the SR backlog was a huge success. The total SR count for the 4 month period peaked at 2584. There existed 1355 SR's at the beginning of the CPE phase. During that time, a total of 1229 SR's were introduced, bringing the total to 2584. During the Q-MIT SR push, a total of 2070 SR's were resolved! Of the 2070 SR's resolved, 462 resulted in some type of fix to MPE. The other 1608 SR's were closed as duplicates, KPR's not yet fixed, user misunderstandings, enhancement requests and transfers. This only left 514 SR's not yet resolved.*

As can be seen by these figures, an impressive number of SR's were closed. But even more important than the total number of SR's closed is the number of fixes that were generated. This effort will eliminate problems in MPE and some of the subsystems. The Q-MIT will now make a strong base for future engineering efforts.

---

*The MPE section was the recipient of the 1982 Software Lab's Quality Award for the significant contribution and improvements made to the quality of MPE software via their CPE efforts.

TOOLS TO ENSURE QUALITY

Privileged Mode Bounds Checking (PMBC)

Changes have been made to the microcode on the Series 64 to support Privileged Mode Bounds Checking. PMBC performs bounds checking on privileged instructions that were previously not checked. More specifically, DB, S, Q and DST relative instructions have added PM checks. Previously, flying data from MOVE STACK/DST type instructions were extremely difficult to catch. PMBC will catch problems in the system before they can cause damage. This will greatly increase the reliability and quality of MPE. PMBC will run only on the Series 64 and will be turned on during all reliability and certification testing at CSY.

Version Control System (VCS)

The Version Control System is a tool developed to aid engineering productivity. VCS supports parallel development efforts and keeps track of the development of software projects. It will be used to track the changes made to MPE modules. VCS will solve the following problems:

- Storage
- Fixes failing to get into multiple versions
- Keeping track of what changes were made and when
- Keeping track of exactly what version a customer has

It is to be used by the software development team to manage their projects and will assist in making the coordination effort planned and systematic.

Logplay

In the past, our interactive menu driven and graphics products could not be tested in an automated fashion. Each time a modification or correction to a product was made, an engineer had to manually test it. This was inefficient and error prone. To improve our ability to test these highly interactive products, a new tool called LOGPLAY was developed. Currently when an engineer creates an interactive test, LOGPLAY can be run concurrently to record the test inputs and results. After program modifications are made, LOGPLAY will automatically send the recorded inputs to the program in an interactive fashion and check the results of this run with the results of the previous run. If discrepancies are detected, messages are printed.

With LOGPLAY comprehensive automated regression tests can be developed for products such as HPDRAW, V/3000, IDS, and DSG. Everytime fixes are made to these products or a new version of MPE is developed these tests can be run and checked automatically with a minimum of engineering time.

SOFTWARE INTEGRATION

SWIMS

SWIMS provides comprehensive control and quality checking of the HP3000 software. SWIMS is a group of user interface packages tied together by a single database. It controls the official version of all software that is on an IT. When a version of code is approved, it is added to the official IT, thus preventing old versions from getting into the MIT. SWIMS also automates the MIT process to insure no undesired variability between MIT versions.

SWIMS provides the engineer easy access to the MIT status, module names with associated owners, module status and history of changes, and an organizational chart of all users of SWIMS. This provides each engineer with the information he needs to make appropriate changes to a module and to check with the owner of the module before each change. SWIMS allows organized access to the software through the following features: the ability to enter changes to any module; the ability to easily print compiled listings of any module; the ability to freeze the MIT at any time; and the ability to identify logically associated changes in different modules/products and guarantee that all of these changes are present.

Every software product has bugs. We realize this and have therefore kept portions of the code open for human intervention, when the process fails. For example: the SWIMS scheduler can be controlled by the Integration group when necessary; the SWIMS software and data files can be moved to any machine quickly in response to machine failures or heavy loads. SWIMS automatically performs, at regular intervals, quality checking throughout the build process; and if there is a global build problem, SWIMS allows the removal of any fix in MPE or any other part of the software under its control. To insure flexibility in each build, the process will be recreated for each build based on information in a database. This allows the MIT contents to be easily changed as required. All file references needed for the build can be verified. thus assuring that all necessary components are present and at their correct version level.

Software Integration/Production Engineering

Software Integration/Production engineering groups have recently been started in CSY and IND. The main role of these groups is to increase quality by coordinating and implementing the standardization of product/inter-product testing, shipment and installation of software. One of the roles of the group in relation to the MIT process is to be the main interface with all other Divisions and Operations involved with the MIT.

B. SOFTWARE INTEGRATION, TESTING AND DISTRIBUTION

A new slogan has been introduced in the MPE lab. It represents a major thrust of quality in the area of software integration and distribution. It is an addition to our past philosophies on improving the quality of our software. The slogan is "EMPHASIZE THE PROCESS OF INTEGRATION AND DISTRIBUTION RATHER THAN JUST THE PRODUCT". What does this mean in terms of quality?

HP has made a commitment to the development of tools to streamline the process of integration and distribution.

LOGISTICS OF SOFTWARE TESTING AND INTEGRATION

A new MIT is developed though a series of passes so that fixes and enhancements can be incorporated in some systematic way. Before an enhancement or fix can be incorporated into the existing software it must meet a set of pre-published "submittal criteria". The submittal criteria are published prior to the start of the integration process and generally refer to the amount of testing that the enhancement has undergone and the documentation that will accompany the enhancement. Through this systematic introduction of new code, any problems encountered can more easily be identified and associated with a particular enhancement.

Starting with the Q-MIT, and for subsequent MIT releases there will be three levels of concurrent testing. The system software has been separated into three distinct categories: MPE (the operating system software), subsystems (e.g. IMAGE, KSAM, etc.) and data comm (the data communication software, e.g. DS/3000, MTS, etc.). The three software categories are tested individually on the existing software. After each software category reaches a certain level of reliability testing, the three are integrated for further testing. This procedure will help in identifying software problems in the initial stages of testing and should facilitate the process of integrating the various categories of software.

Finally, when the integrated software reaches an acceptable level of reliability testing, it is installed on various internal systems as "Alpha sites". These alpha sites are monitored closely to uncover any problems which the reliability testing might not have uncovered. As part of the test plan for the MIT testing procedures, an Alpha Test Plan is developed by which to measure the progress of the testing and establish goals for the testing.

SOFTWARE TESTING

Within the normal development of a MIT, three types of testing will be used for the entire process. Much of the testing is automated and will continue to be automated in the futere. The primary objective of testing is to exercise all the paths within a software product. To locate and test all the paths within the software is not a trivial task and is often related to complex static and dynamic analyses. Past experiences suggest that without the aid of testing tools, tests generated by humans cover less than 50% of program paths.

Formalized Test Plan

The Q-MIT and all subsequent software releases will be governed by a formalized test plan. Every software release for the HP 3000 undergoes a long process of software integration and testing. Criteria for the testing process has, in the past, been left up to the product team (members of the various functional areas involved with the product). As a result, each product or MIT has been tested under different standards and evaluated under a different set of criteria. A formalized test plan is currently being developed to give the lab more direction and to establish a consistent level of quality in our software products.

Throughout the duration of a MIT project, the software is "built" and tested several times. The MIT is tested though a series of builds called passes. Each pass of the software must undergo the planned reliability testing outlined in the test plan before the next pass can be developed. Between each pass there can be a number of builds and testing cycles to meet the desired reliability testing outlined in the test plan.

The new test plan sets concrete objectives for each pass of the software so that the product team can properly evaluate the success of a build. Resource utilization and reliability testing time is also outlined for each build, which allows the software integration group to accurately determine the length of time for each test.

This new test plan helps to optimize the test process, it organizes the resources and gives the engineers doing the testing clear and concise procedures to follow. This plan guarantees a certain level of quality for each pass and aids in the creation of realistic schedules. By formalizing the testing process, we have provided a systematic way of evaluating the quality of the product and provided a measure by which future products can be equally evaluated.

Software Tests

Automated Reliability - This test is a series of self streaming jobs executing on four different systems. These four systems, a Series III, a Series 33, a Series 44 and a Series 64, have different peripheral configurations and are linked together with DS/3000 lines. The intent behind this test is to put a heavy workload on the system and test as much of the functionality of the system as possible. By putting such a load on the system, this test uncovers programming errors and timing problems within the code. Additionally, this test is flexible enough so that the jobs can be changed to test specific sections of the software.

Certification Testing - The certification tests are designed to test MPE and HP3000 subsystems. The tests are to be updated and enhanced in coordination with the ongoing development of HP software. Previously, all of these tests required operators to initiate and monitor the tests as they ran, and then evaluate the results manually. In addition, some tests required extensive operator intervention. As testing was often run 24 hours a day and through weekends, engineers and operators who worked late hours and extra shifts understandably introduced human errors into the testing process. This had the effect of reducing productivity and caused much energy to be devoted to running tests instead of solving problems and developing new features.

In an effort to maximize productivity and improve the turnaround time for test evaluation, Software QA has developed the AUTOMATED CERTIFICATION TEST UMBRELLA. This represents an effort to automate the running and analyzing of all of the certification tests. Approximately 83% (24/29) of these tests were automated in time for QMIT testing. The Umbrella ran all of the tests and then compared the results against a benchmark of what the results should be. In this way, the Umbrella provided a quick and accurate evaluation of the state of MPE from the early stages of development through the final testing.

Path Flow Analyzer (PFA) - The Path Flow Analyzer is an automated tool used to maximize the effectiveness of testing. Essentially, the PFA is intended to be applied during program testing to identify untested paths and to aid in specifying test cases that will improve testing coverage. All of this is provided by analysis of program structure, instrumentation of the software with probes that measure testing coverage, and generation of comprehensive reports which pinpoint paths in the program structure that remain to be exercised. In addition, guidance is provided for the generation of test cases that will assure coverage of the untested portions.

SOFTWARE DISTRIBUTION

An integral part of the overall perception of software quality is the quality portrayed by the software distribution mechanisms. To insure that customers perceive high quality in our distribution system, software must be readily installable through a reliable process when it arrives at the customer site.

MITWARE

For Q-MIT, and perhaps the next few software releases, MITWARE will be supplied with the software for the Field Software Coordinators (FSC). MITWARE contains a database of all Hewlett-Packard products and the various modules of MPE. A customized tape can be created with MITWARE by "picking" those products that the customer is entitled to have and integrating them on the system tape. With MITWARE, the FSC or SE can easily create a customer installable tape. Additionally, because MITWARE has the MPE modules stored in a database, a customized tape for any system can be created from any other system.

Direct Distribution

Our direction is direct distribution of software to customers. With this procedure there would be no need for system customization; rather, the system would retrieve only those products the customer is entitled to have. This scheme would offer many benefits; for instance, any errors normally introduced during the customization stage would be eliminated.

A requirement for direct distribution would be a centralized software distribution center. The "master" system tape would be duplicated, under controlled conditions, for mass distribution. There are many advantages to centralized distribution. By centralizing the tape duplication function, the process can be controlled and streamlined for better efficiencies. Additionally, the process can be tracked and studied to constantly improve the quality of the software being released to customers.

CUSTOMER FEEDBACK -- STARS II

Hewlett-Packard's most effective means of relaying customer feedback to the factory has been through the Service request Tracking and Reporting System (STARS). With the Q-MIT, a new system of service request tracking is being introduced: STARS II. This new system will expand the functionality of the old STARS facility, and will vastly improve the efficiency of reporting and fixing bugs.

The old STARS facility made extensive use of paper forms for reporting bugs and following fix solutions through various stages. The new STARS II is much more "on-line." Eventually, many area offices will be connected to a STARS II computer. Various engineers (SE's, lab, CE's, marketing engineers, etc.) now access the STARS II data base directly, virtually eliminating the proliferation of forms and papers which now exists.

With STARS II, SE's will enter service requests to an SR Transaction file which will be sent via the Computer Support Division (CSD) to CSY. The service requests in this file will be merged into the STARS II data base. SE's will be able to retrieve information directly from the STARS II data base. This system will relieve the SE's of the paperwork associated with submitting service requests.

SEO field Phone-in Consulting Service (PICS) centers will benefit by being able to access more current information than what is in the SSB. This will allow the field to respond better to customer problems when they have been previously reported and should siginificantly reduce the duplication of effort to develop "workarounds" for those previously reported problems.

STARS II will help cut down on the duplication of Known Problem Reports (KPR's). Software problems will be more easily classified and located in the data base, providing an efficient, on-line alternative to the Software Status Bulletin (SSB).

This new software tracking system will be more streamlined and efficient than what has been available in the past, permitting SE's, CE's, On-line Support, and lab engineers to spend more time doing actual constructive work, and less time shuffling paper.

C. SOFTWARE DOCUMENTATION

With Q-MIT, several improvements have been made in the area of user documentation for MPE. The most notable changes have been the transfer of the user manuals to on-line status, the updating and upgrading of several specific manuals, and the improved distribution of manuals.

ON-LINE MANUALS

In the past, documentation materials were typed or hand-written, and then typeset on art boards by a separate graphics department. Today, as part of the push toward greater quality, the MPE user manuals are being placed on-line. All new materials (new manuals, update packages) are written on-line from their conception.

Documentation materials are entered using TDP/3000, and printed on the 2680A Page Printer. Printed material is of comparable quality to typeset art boards, and may be sent directly to the printer for processing. Since the writers have the proper tools for producing their manuals, the need to go through the typesetting departments has been eliminated and with it the numerous errors introduced.

Not only are the manuals themselves being placed on-line, but the process of generating a Table of Contents and Index for a manual has been automated. This saves considerable time that would be spent checking page numbers and supervising typing and layout, and reduces the possibility of human error. Additionally, by having the manuals on-line the spelling of the text can be verified through a program that detects mis-spelled words.

Since the process of producing documentation materials has been streamlined, and since superfluous interfaces with other groups have been eliminated, writers can spend more of their time doing actual researching and writing. The relieved time pressures allow the writers to do a more thorough job, and the increased efficiency results in reduced cost, higher quality, and better technical accuracy.

IMPROVED MANUAL DISTRIBUTION

The MPE documentation group is currently working with representatives from the Software Distribution Center to improve the process of manual distribution. Specifically, the goal is to coordinate manual distribution more closely with the release of software products

D. QUALITY CONSCIOUSNESS OF EACH ORGANIZATION

A major goal within BCG is to help the divisions improve quality by making it an integral part of the HP culture. This is accomplished by presenting quality-related programs and teaching the methodology associated with achieving quality. These programs include classes, workshops, seminars, publications and quality circles.

CLASSES

Software at HP

The Software at HP class is a joint offering of the CSY Software Training Group in the lab and the Quality Improvement Program in Product Assurance. It is designed to familiarize new Business Computer Group engineers with the software development process. Students are introduced to tools like the Software Product Life Cycle (SPLC). Various phases and aspects of the SPLC are discussed in detail. The participants also learn what goes on in the field to gain a better appreciation of how their work affects the work of the service organizations. Other topics covered include the MIT Build Process, Cost of Quality, testing, the function of Product Teams, and finally, a management panel.

WORKSHOPS

Marketing Quality Day

On August 13th, 1982, CSY Product Assurance hosted one of the most unusual and successful quality-related events in recent memory. It was entitled "Quality: A Key Marketing Issue For The 80's" and was presented to the Marketing Departments of BCG over 250 people attended.

After morning presentations the afternoon session involved dividing participants into six workshops. The workshops were designed to give people the chance to explore various topics in depth:

CUSTOMER NEEDS - How can Marketing do a better job of letting R&D know what customers want?

CUSTOMER SERVICES - How to improve the quality of:

* setting expectations;
* product documentation;
* advertising;
* promotion;
* customer training;
* field training.

COMMUNICATION - As departments, divisions and groups, how can we understand each other better?

PRODUCT DESIGN - How can Marketing develop a well integrated, clear process from marketing research through support? (What is Marketing's role in the Product Life Cycle?)

SCHEDULING - How can Marketing assist in the timely and professional release of new products?

QUALITY ADVANTAGE - How can Marketing unleash HP's hidden weapon: Quality?

The day concluded with workshop leaders making brief presentations to a panel of Marketing managers about what their groups discussed. The managers commented on the presentations and exchanged ideas with the audience.

SEMINARS

Seminars have been given on many topics - control charts, path flow analyzer, static electricity and vendor management, three seminars stand out as unique.

HP Labs Seminar Series

This series was created to share with BCG hardware and software engineers the technology and projects being developed at HP Labs in Palo Alto for products that will be introduced five to ten years in the future. To date, presentations have been given on artificial intelligence, computer architecture, manufacturer's productivity network, and robotics - over 1000 people attended these four seminars.

LACE Seminars

The object of the LACE series is to sensitize Lab engineers and other HP personnel to the environments in which the systems they design are actually being used. The customer's viewpoint of our hardware and software is the focus of these seminars.

Customers are invited to CSY to share information concerning different aspects of their companies and computer systems:

company products and services
primary applications
operating environment
systems management philosophy
volume (#users, jobs, sessions, printed pages...)
hardware configuration
HP software used
'wish list' of features

Questions and answers are an important part of these sessions.

It is clear from the positive feedback and high attendance that the LACE seminars are valuable to both the attendees and the customers. The following customers have shared in our LACE program:

* Austin Information Systems
* Lucasfilms
* Weyerhaeuser Company
* Long's Drugs
* Information Systems Council

IBM Design Inspection Seminar

On January 10th and 11th, 1983, CSY Software Product Assurance hosted a two day seminar presented to the HP R&D engineers and managers by an IBM Quality Assurance Specialist. The seminar was attended by over 100 engineers and managers, representing twelve software divisions at Hewlett Packard. One day was addressed to managers only - if you don't get management support for new techniques, they won't succeed in any environment. The seminar was a 'how to' session on the use of Design Inspections to improve the productivity and quality of software products.

The materials and enthusiasm generated by the two day seminar have laid the groundwork for several classes - including one in the planning stages at CSY. The idea is to further transfer the techniques to all engineers in the lab organization, making higher quality products through smarter technology.

The seminar is a good example of how Hewlett Packard is participating with other companies to explore and improve the technology and quality of software. HP is playing a leadership role in generating the standards of software quality for the future, with the Co-chairmanship of the IEEE Software Reliability Measurement Industry Taskforce residing at CSY (shared with IBM).

QUALITY CIRCLES

Quality Teams are Hewlett-Packards's version of Japanese Quality Circles. A Quality Team, or Circle, is a group of three to fifteen employees from the same work area who voluntarily meet for an hour to identify, analyze and develop solutions to work-related problems. These solutions are then recommended to management in a presentation format.

CSY has twenty Quality Teams up and running - six in the software lab. The most notable project resulting from the Quality Teams in the software lab is a package of structured programming guidelines which will become the standard for future software development in CSY's lab.

HP Quality Teams are one effective way for the organization to involve its employees in assuming responsibility for quality... the quality of their work, their work environment, their professional growth, and their personal development.

CONCLUSION

Major improvements have been made in quality over the last year, but we are going to have to be very innovative and creative to stay ahead of the competition. Quality is everybody's responsibility. Each person has a role to play if we are to develop high quality products that are price competitive.

Through the quality activities identified in this presentation* we are actively working to build awareness of our "quality challenge" in the minds of everybody in the Business Computer Group. It is only through team effort and a unified commitment to quality that we will be able to maintain and build upon our current leadership position.

To paraphrase a modern day Japanese slogan: "The battlefield of the 80's is Quality."

---

*My thanks to Sam Quezada, HP CSY Marketing

IMAGE/3000 PERFORMANCE PLANNING AND TESTING

Walter W. Gioscia
Data Base Administrator
The Bovaird Supply Company
Tulsa, Oklahoma

Dennis Heidner
Boeing Aerospace Company
Seattle, Washington

INTRODUCTION

A data base management system (DBMS) provides a framework for handling files and groups of information. The file system is a rudimentary DBMS since it provides the capability of structuring and maintaining sets of information. A DBMS extends the capabilities of the file system by specifying file relationships, field contents, and field security. It also provides shared access and locking of data entities.

In data base environments the most common issue is performance. What is performance? How do you measure it? Can you predict it? In order to understand the service provided by such data base management systems as IMAGE you need a performance monitor which collects and analyzes data in your working enviornment. Such tools are available from the contributed library (SCANUSER, FILERPT, LOGLIST, DBLOADNG, DBSTAT2) and from a number of vendors. Hewlett Packard (HP) provides a tool called OPT3000 which provides information on the system as a whole. Unfortunately until you have a great deal of experience it is difficult to interpret the data accurately. It is even harder to use your data to predict future performance.

What is needed is some baseline data so that you can compare your system against a known system. Once this is done you can more accurately tailor your system to your application. Still needed however, is a method to estimate the performance of new data base applications, guidelines will be developed in this paper.

The IMAGE DBMS is a network of closely knit MPE files. IMAGE protects the data base for ordinary users by designating the files as privileged. Access to information in the data base is provided by calling IMAGE procedures. IMAGE uses special tables to knit the files together. If we carefully read the reference manuals we find:

"A Data Base Control Block (DBCB) is a table of data base relative information residing in a privileged extra data segment. There is exactly ONE DBCB for each open data base regardless of the number of concurrent access paths to the data base. The DBCB contains all static and dynamic global information; that is, it contains all information which is not unique to a particular access path." [1]

"In addition to tables describing the various components of the data base, the DBCB contains all buffers and work areas used by the IMAGE procedures. IMAGE may modify the size and contents of the various areas within the DBCB to reflect the addition of access paths as more users open the data base. All IMAGE procedures operating on a particular data base reference the same DBCB." [2]

"All buffers whose contents have been changed to reflect a modification of the data base are always written to disc before the library procedure exits to the calling program." [3]

It should be apparent that the DBCB is extremely important to the operation of the IMAGE. What is not told, however is how important the DBCB is to the performance of IMAGE data bases!.

IMAGE intrinsics generally reference the DBCB in unpredictable patterns. In order to maintain the integrity of this important resource, IMAGE prevents other processes from using the DBCB when it is already in use by maintaining a waiting list or queue. Processes waiting in line are served strictly on a First Come First Serviced (FCFS) basis.

The fact that IMAGE uses a FCFS type of queueing method makes it very convenient to understand, model, and estimate waiting, response and transaction times.

IMAGE and QUEUES

The classic example of a queue is a bank teller and the banks customers. With only one teller (server) only one customer may be handled at a time. If the teller requires one hour per transaction and the customers arrive once an hour, then as a rule there should never be any waiting time. However, if we now double the number of customers so that they arrive once every half hour, and the bank teller still takes one hour per customer, then by lunch time there will be approximately four customers in the waiting room waiting for help! The last customer would depart four hours after arrival. (If they are smart, they would also change banks ...)

Of course, the above example over simplifies the problem. In real life you would not expect to have arrivals only once per hour, but in a more random fashion. Experience has shown that the typical arrival rate follows what is known as a Poisson distribution. Thus to find the PROBABILITY that N customers will arrive within given time T, we use the following formula:

The probability that n customers would arrive is: [4]

$$P = \frac{(L)^n \, e^{-L}}{n!}$$

T = time interval.

n = number of customers

e = Euler number (2.718)

lambda = the usual arrival rate (or mean arrival rate) The mean arrival rate is the average number of customers who appear in a given time.

L = T*(lambda) This is the time interval in question multiplied by the normal arrival rate.

In the bank teller example we said the usual arrival rate (mean arrival) was one hour. If we are interested in the probability of customers arrivin during a 15 minue period then we calculate:

L = .25 hour * (1 / hour) = .25

Let's try some numbers ...

$$P = .25^{1} e^{-.25}$$

$$P = .194$$

Probability for 1 customer during the 15 minutes is

P = .194 / 1 = .194 or about 1 in 5

Probability for 2 customers during the 15 minutes is

P = (.25)(.194)/2 = .024 or about 1 in 40!

The probability that 10 customers would arrive during the 15 minutes is:

$$P = \frac{.25^{10} e^{-.25}}{10!}$$

P = .000000000000020467 or 1 in 488 billion!

In our simple example we declared that each customer required one hour of the teller's time. In real life however, we could expect that there might be some average service time. We will call it Ts. If we multiply the arrival rate (lambda) and the service time (Ts) we get a figure called the traffic intensity, P. When the traffic intensity (P) is greater than one, then the system is OVERLOADED. P is also an indication of the number of servers that would be required to prevent the growth of long waiting lines. If P was 3.9, you would need four servers. In this case, if the customers are IMAGE and the DBCB is the server, four DBCB's would be needed to handle the arrival rate and service times.

Our real interest is in how long the customers will wait. (Perhaps we could set up a coffee shop in the lobby!)

Please note that the 'waiting time' is not the 'response time'. The definition of response time is the waiting time PLUS the time to complete the work. In other words, if there is no waiting time but it takes 1 hour for each customer then the response time is 1 hour. On the other hand if the waiting time is 59 minutes and the service time is 1 minute, the response time is still one hour! (It should now be obvious why long IMAGE transaction times cause a lot of problems on the computer! In many cases the time to perform the IMAGE transaction is the driving factor in the response time.)

From the book SOFTWARE SYSTEMS PRINCIPLES, A SURVEY by Peter Freeman, [5] we obtain the following formula for mean waiting time and the variance of the waiting time, based on Poisson distributions.

The mean waiting time W is:

$$W = \frac{(\text{lambda})(Ts)^2}{2\,(1 - P)}$$

lambda = mean arrival rate.

Ts = mean service team.

P = traffic intensity. P = lambda * Ts

Note that as the traffic intensity approaches one, the waiting time goes out of sight. In other words, as the IMAGE data base control block (DBCB) approaches 100% utilization, the response time seen by the user skyrockets!

The variance, V, is the measure of the spread of the waiting times around the average. The larger the value for V, the more erratic the computer response time appears to its customers. The variance of the waiting time is:

$$Y = \frac{(\text{lambda})(Ts)^3}{3\,(1 - P)} + \frac{[\,(\text{lambda})(Ts)^2\,]^2}{4\,(1 - P)^2}$$

Applying the theory:

Let's assume we have a data base called XYZ. We know from experience that the average transaction takes 1.6 seconds. We also know that during an eight hour day we have 960 of these transactions. What would you expect for the response time, waiting time, variance, and probability of three transactions within thirty seconds?

The first step... Let's convert the transaction interval into transactions per second.

$$\text{lambda} = \frac{960 \text{ transaction per day}}{480 \text{ minutes per day}} \times \frac{\text{minutes}}{60 \text{ seconds}}$$

lambda = 1 transaction per 30 seconds
= .033 transaction per second

The average service time, Ts, was specified as 1.6 seconds.

The trasffic intensity, P, is then:

$$P = (1.6)(.033)$$

$$P = .053$$

Notice that at present we are far from being overloaded!

The waiting time, W, is:

$$W = \frac{(.033)(1.6)^2}{2\,(1 - .053)}$$

$$W = .044 \text{ SECONDS}$$

The response time is the mean service time plus the waiting time, or:

Response time = W + ts = .044 + 1.6 = 1.644 seconds.

NOTE!!! The dominant factor is not the volume of work, but the time required to make the individual change in the IMAGE data base!

The variance, V, is:

$$V = \frac{(.033)(1.6)^3}{3\,(.946)} + \frac{[\,(.033)(1.6)^2\,]^2}{4\,(.946)\,(.946)}$$

$$V = .030 + .002$$

$$V = .032 \text{ SECONDS VARIANCE}$$

Approximately 90% of the transactions are covered in the mean waiting time PLUS the variance. In other words, for all practical purposes, all transactions will take between 1.6 and 1.7 seconds!

In order to obtain estimates for the average service time we must run tests on IMAGE. One method of testing is called benchmarking. A benchmark is a program that applies a known workload to the item under test in a controlled environment. The benchmarks that we have developed were designed to push IMAGE and the hardware that supports it to their upper limits.

TEST ENVIRONMENT

During the course of our testing we considered the following variables that could affect IMAGE performance:

- Hardware and system configuration -
  The benchmarks were run on four different systems. These systems varied from a 1 MB series 33 to an 8 MB series 64 as shown in figures 1 thru 4. Test were run using IMAGE version B.02.XX.

- Number of concurrent processes -
  The number of processes running was varied from 1 to 60.

- Process characteristics -
  Stack and code segment sizes were kept at a minimum and all processes were run with a priority of CS.

- CPU utilization -
  Tests were run to see what effect CPU-bound processes would have on IMAGE transaction throughput (I/O)

- File location on the disc drives -
  Tests were run to determine the effects of carefully selecting the disc drives when creating data bases. This test involved two competing data bases.

- User 'think time' -
  The user 'think time' is the inverse of the arrival rate (lambda). We did not induce any 'think time'. The purpose of the test was to push IMAGE, not MPE, to the limit.

- IMAGE transaction logging -
  All tests were done without and with transaction logging enabled.

- Multiple data bases -
  Tests were run to determine what effect additional data bases would cause.

The benchmark programs used were written in FORTRAN. They were DBPERF, DBPERF2, DUMYLOAD, DUMY2, CPULOAD, and CPULOAD2. Two test data bases ABC and XYZ were used. The data bases had identical structures. [6]

The programs DBPERF and DBPERF2 can create and activate up to 80 additional processes running in the same priority queue. DBPERF and DBPERF2 are the test and data collection programs. [7]

The DUMYLOAD and DUMY2 programs try to perform 10,000 iterations of DBFIND, DBGET, DBLOCK, DBUPDATE and DBUNLOCK. [8]

The programs that were used to load only the CPU are called CPULOAD and CPULOAD2. These programs perform multiplications, data moves and a large number of internal segment PCALS. [9]

TEST RESULTS

Charts 1 and 2 compare the times required to perform DBPUT's and DBDELETE's on data base XYZ. The tests were run without and with transaction logging enabled. As can be seen on the charts, the overhead added by transaction logging is insignificant. As we repeated the tests with 10, 20, 40 and 60 DUMYLOAD the overhead was nearly the same. Because the performance without and with logging is so close, we did not plot the results of transaction logging on any of following charts.

Chart 3 shows a comparison of the time to perform DBOPEN's, DBUPDATE's and DBBEGIN-DBEND pairs on all the tested machines.

Charts 4 thru 11 demonstrate the effect of increasing the number of processes accessing the XYZ data base. It should be noted that the slight increase in transaction service time appears to be due to increased processing that IMAGE does to maintain the internal I/O buffers and the who's-next-accessor list. To estimate the transaction response time that users will experience, you may obtain values for Ts from the charts and use the queueing formula presented previously.

Another phenomenon that can be seen in charts 4 thru 13 is that it appears that the series 64 system is slower than the III or 44. Since each drive on the 64 was a master, on separate GIC's, it would appear that the 7933 disc drives are slower than 7920 and 7925's.

Chart 12 shows the effects of data bases ABC and XYZ competing for system resources such as disc I/O and CPU time. An important observation is that in this test the data bases were created only with DBUTIL.

Chart 13 shows the effects of data base ABC and XYZ competing for system resources, but this time with files spread so that the files for ABC and XYZ are on different drives. Note the dramatic time improvement for the series III computer.

Chart 14 demonstrates the effect of loading the CPU with a process that consumes 97% of the available CPU time. Fortunately in most environments the need for CPU time comes in short bursts. Table 1 lists the total elapsed times for the tests and the CPU time used. An interesting observation is that although a test on the series 44 used 50% more CPU time as the series 64, the total elapsed time was 1 minute shorter.

PREDICTING PERFORMANCE

Real life differs from our tests in a number of important ways. First the work load for most computers can be broken into distinct shifts and/or subperiods. For instance, you might have a large crew entering data during the day and a second smaller crew correcting and examining data at night. It is even likely that you could break your normal day shift into smaller periods such as early morning before coffee break, mid-morning before lunch, early afternoon, and late afternoon. The response times, waiting times and variances should be calculated for each period individually.

Second, there will generally be several different programs accessing the data base. To model the system correctly you must estimate the required service time for each process, and then estimate the percentage of time that that process will be running during the various work periods noted above.

Third, not all programs will be making actual data base transactions; many applications have terminals and processes which only are used to scan the data base. If these processes are I/O or CPU intensive, then they can play an important factor in determining the system response time. It should also be noted that in order to perform a DBDELETE or DBUPDATE the entry must first be found and read. It is important to include this in the calculation of transaction service times.

Last, there will probably be other non-IMAGE CPU and I/O activity that needs to be considered, such as program development or maintenance.

SUGGESTIONS FOR IMPROVING IMAGE PERFORMANCE

If an application makes heavy use of data bases, careful consideration should be given to an appropriate data base design. As the charts show, each path into a data set is expensive.

If possible, try to build a smaller test version of your spplication data base, then benchmark it. Make sure all your analyst and programmers are aware of path time costs.

Assign adequate capacities to the MASTER data sets. Be sure to use PRIME numbers for MASTERS. If you ignore these simple rules you will see an increase in the number of synonyms in your data base which will severely affect your data base performance.

If program development and maintenance consumes a large amount of CPU time and disc I/O, consider separating these functions from your production applications, and put them on their own computer.

Check your data base for loading problems with DBLOADNG [10] or HOWMESSY [11].

Reduce disc head contention between data bases by carefully chosing the locations for your data base files, see charts 12 and 13.

If you have a series III and appear to be I/O bound, then be sure to use 'seek ahead'. If you have a series 44 or 64 then add additional GICs and convert as many drives as you can to masters. [12]

Avoid mixing heavy CPU bound applications with heavy IMAGE applications, as can be seen in chart 14 they don't mix well.

The importance of scheduling your workload can not be overstated. You can see from the charts and the prior discussion on queueing that the greater the number of processes trying to access the data base the longer you must wait. If possible all online changes should be kept as short as possible. Reports should only be produced during second or third shifts.

AREAS FOR FURTHER INVESTIGATION

There are a number of IMAGE performance topics not covered by our benchmarks. These include the effects of sort items, manual master versus automatic masters, buffspecs, blockmax and choice of data types. These may be covered in future tests.

CONCLUSIONS

The benchmark tests run have provided the users with baseline information for evaluating the performance of IMAGE. This data, combined with the knowldge that IMAGE uses a FCFS type queueing, permits us to estimate the performance of future applications. We have also learned that the cost of adding additional search items to data sets is significant.

Remember to use the formulas presented to help you estimate waiting, response and transaction times.

CONFIGURATION OF SYSTEMS

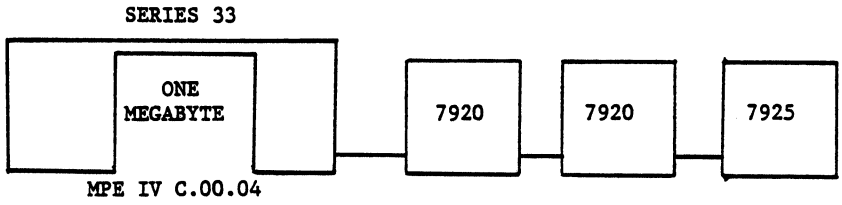


FIGURE 1

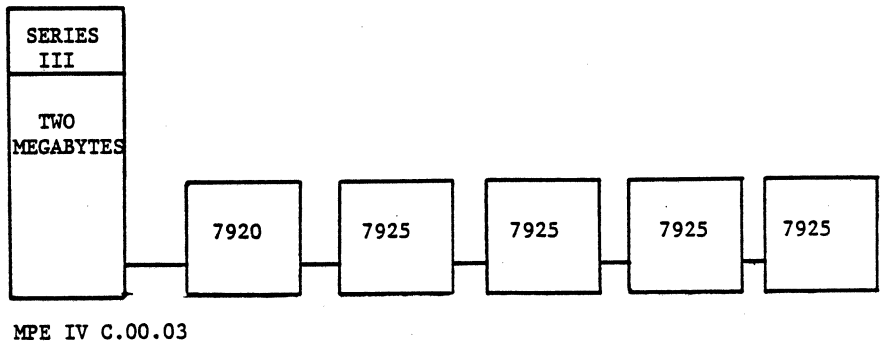


FIGURE 2

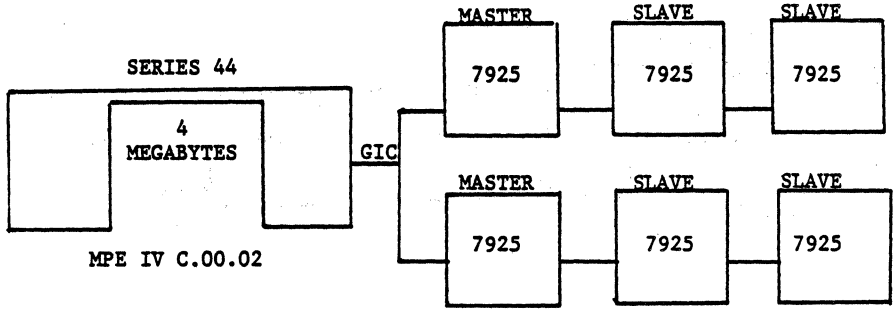


FIGURE 3

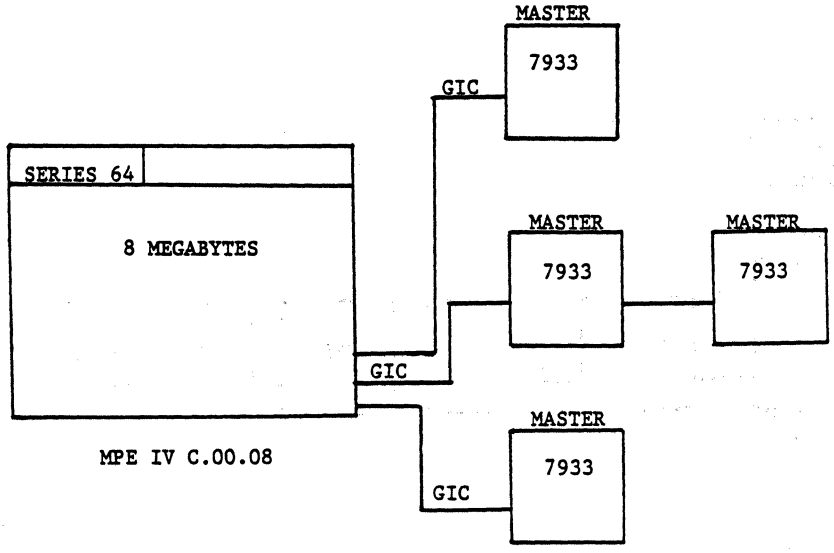


FIGURE 4

IMAGE-3000 BENCHMARK RESULTS

DBPUT's to XYZ with only DBPERF running

1 - 33 without logging 3 - III without logging 5 - 44 without logging 7 - 64 without logging
2 - 33 with logging 4 - III with logging 6 - 44 with logging 8 - 64 with logging

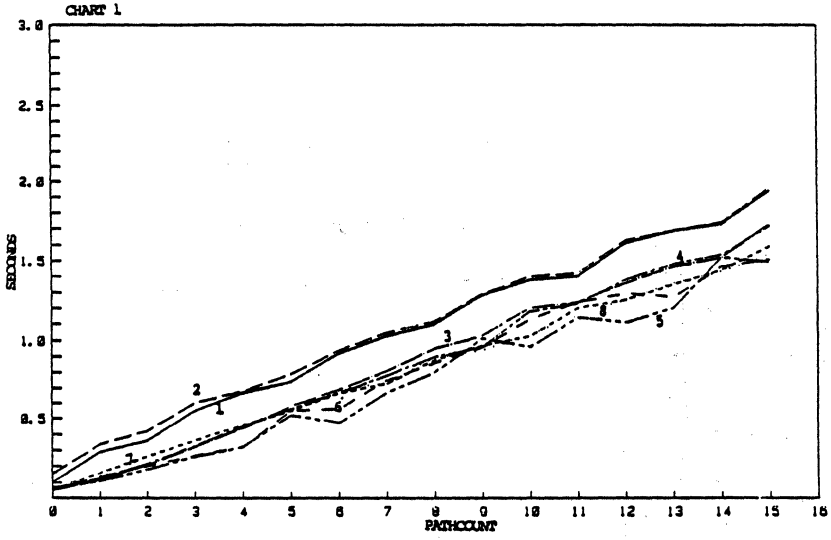
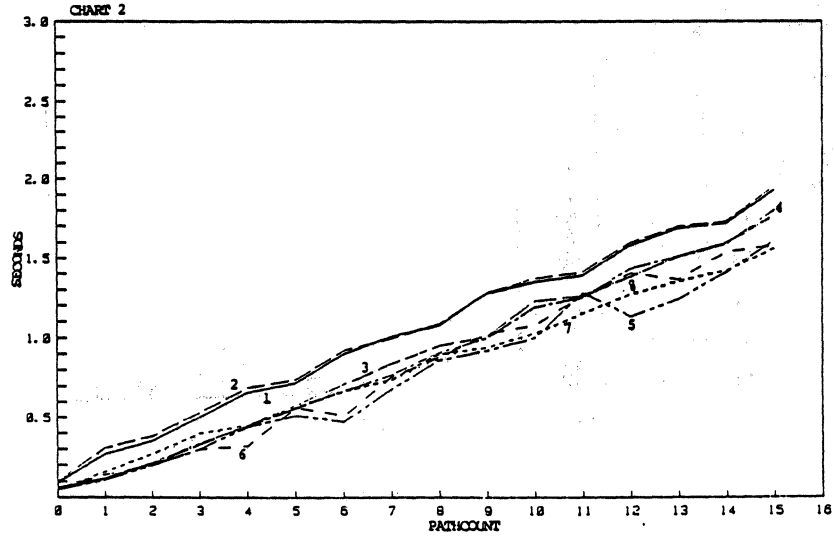


IMAGE-3000 BENCHMARK RESULTS

DBDELETE's to XYZ with only DBPERF running

1 - 33 without logging 3 - III without logging 5 - 44 without logging 7 - 64 without logging
2 - 33 with logging 4 - III with logging 6 - 44 with logging 8 - 64 with logging



IMAGE/3000 BENCHMARK RESULTS

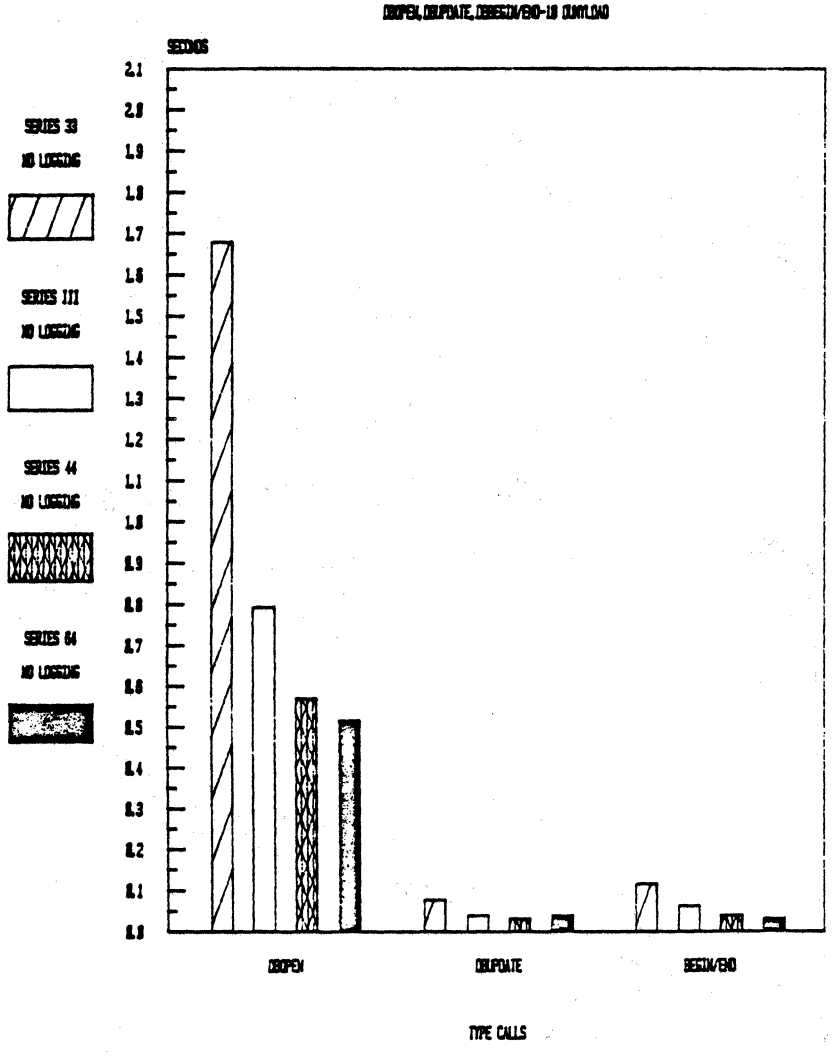


CHART 3

IMAGE-3000 BENCHMARK RESULTS

DBPUT'e with 10 DUMYLOAD

33 WITHOUT LOGGING — III WITHOUT LOGGING — 44 WITHOUT LOGGING — 64 WITHOUT LOGGING

CHART 4

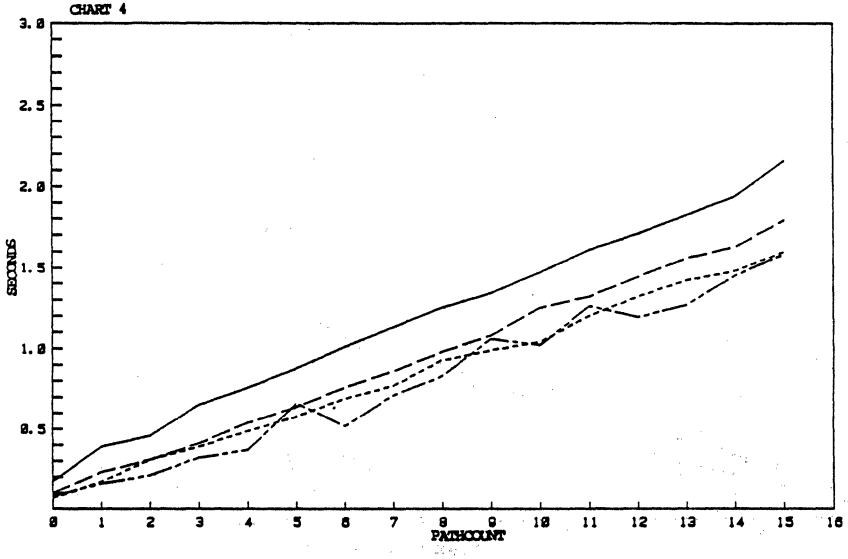


IMAGE-3000 BENCHMARK RESULTS

DBDELETE'e with 10 DUMYLOAD

33 WITHOUT LOGGING — III WITHOUT LOGGING — 44 WITHOUT LOGGING — 64 WITHOUT LOGGING

CHART 5

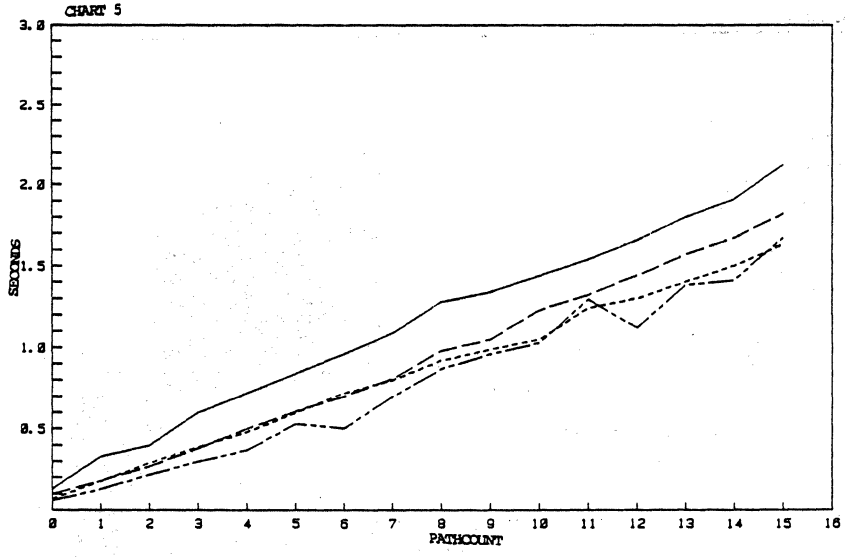


IMAGE-3000 BENCHMARK RESULTS

DBPUT's with 20 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING --------

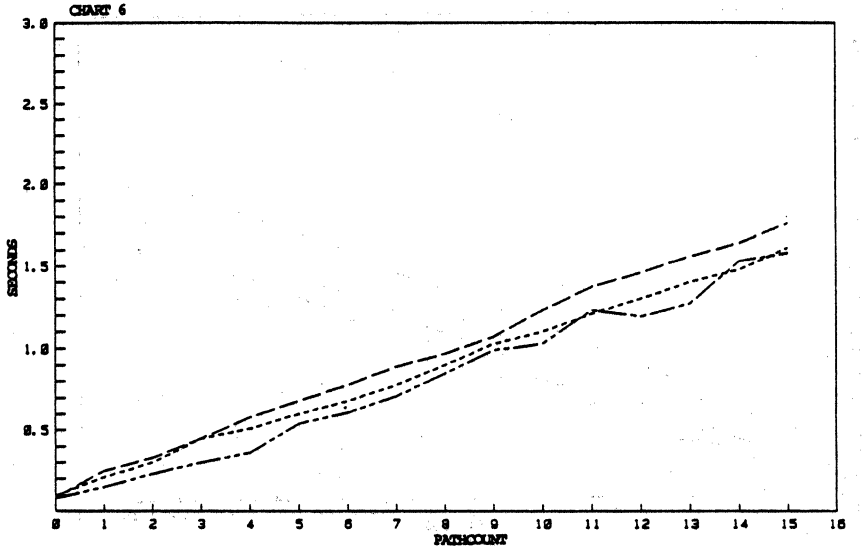


IMAGE-3000 BENCHMARK RESULTS

DBDELETE's with 20 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING --------

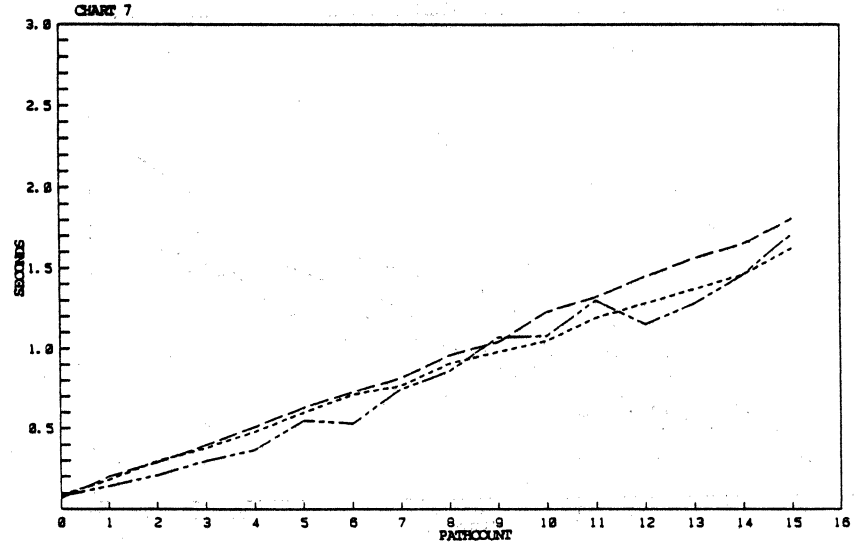


IMAGE-3000 BENCHMARK RESULTS

DBPUT'e with 40 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING ---------

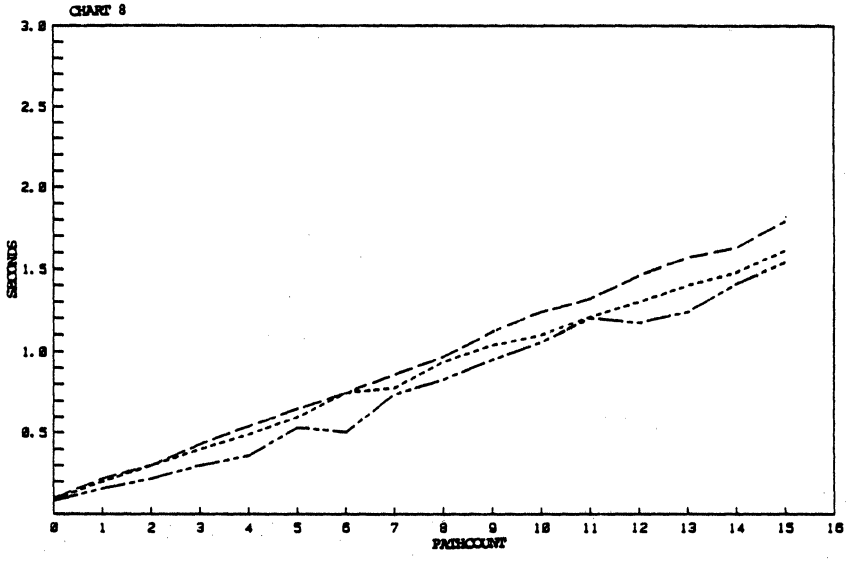


IMAGE-3000 BENCHMARK RESULTS

DBDELETE'e with 40 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING ---------

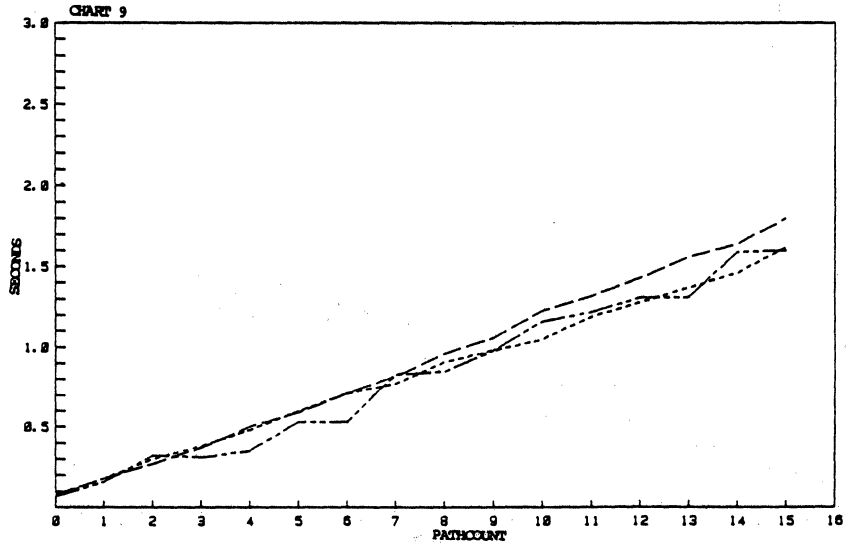


IMAGE-3000 BENCHMARK RESULTS

DBPUT's with 60 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING ——·——
64 WITHOUT LOGGING --------

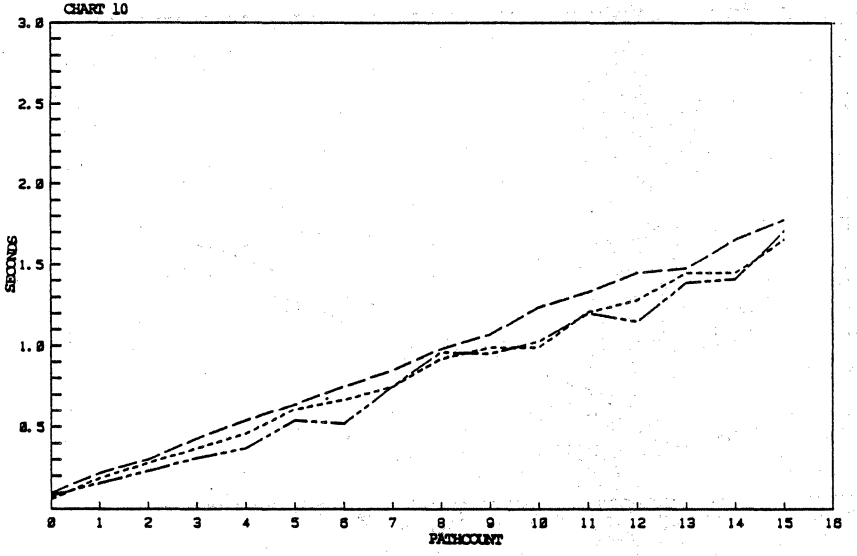


IMAGE-3000 BENCHMARK RESULTS

DBDELETE's with 60 DUMYLOAD

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING ——·——
64 WITHOUT LOGGING --------

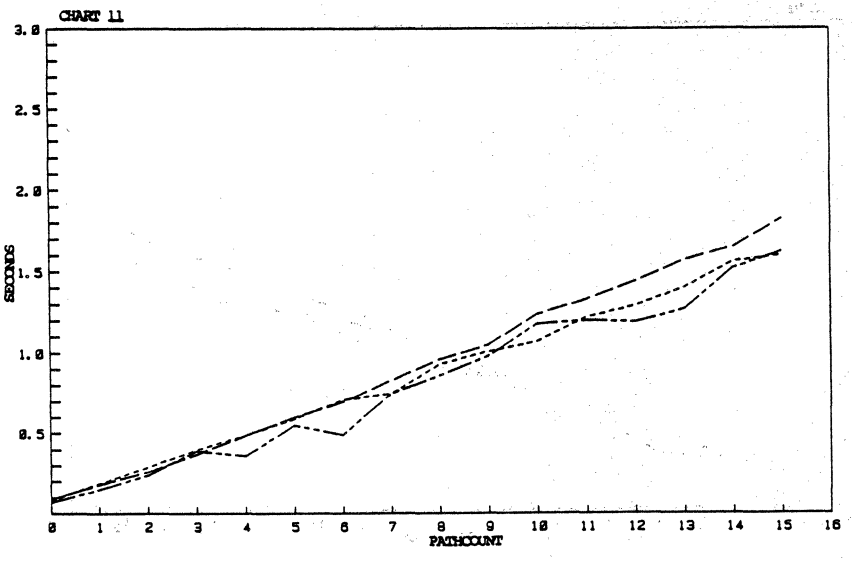


IMAGE-3000 BENCHMARK RESULTS

DBPUT'S for XYZ and ABC - 20 DUMYLOAD each (DBUTIL CREATED)

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING ---------

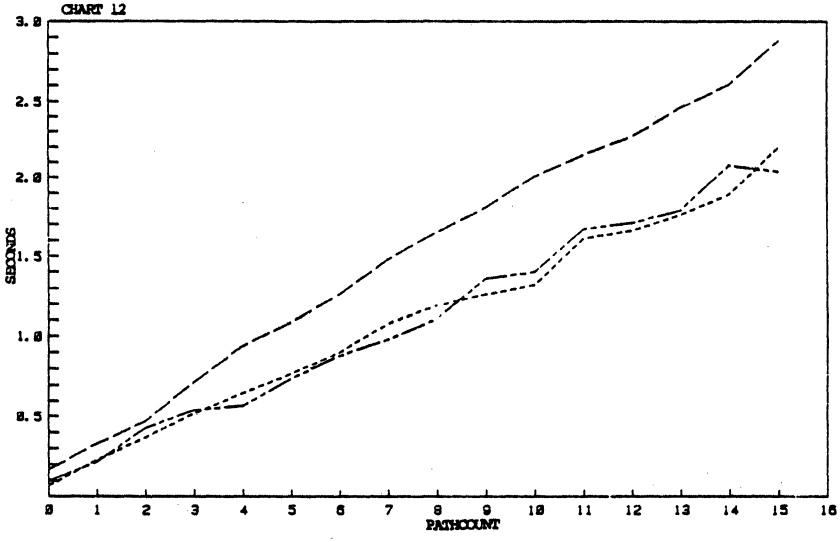


IMAGE-3000 BENCHMARK RESULTS

DBPUT's for XYZ and ABC - 20 DUMYLOAD each (MANUALLY SPREAD)

III WITHOUT LOGGING — — —
44 WITHOUT LOGGING —·—·—
64 WITHOUT LOGGING ---------

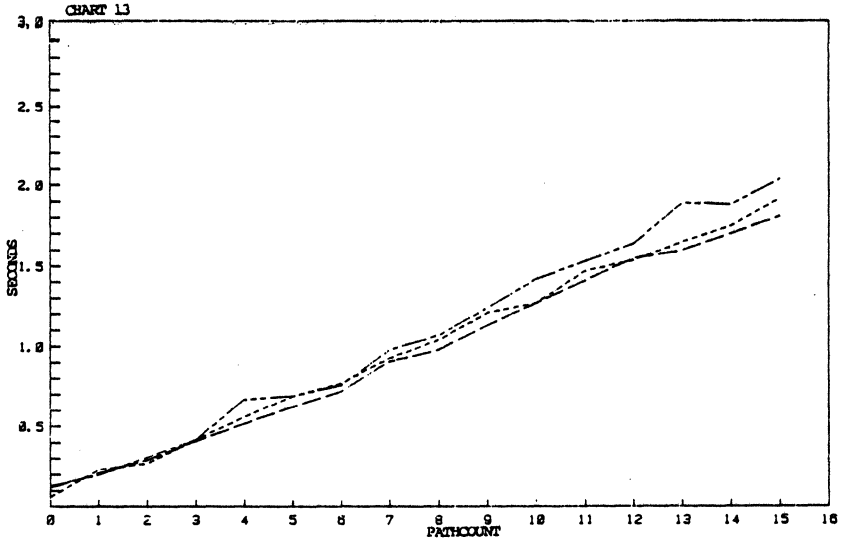


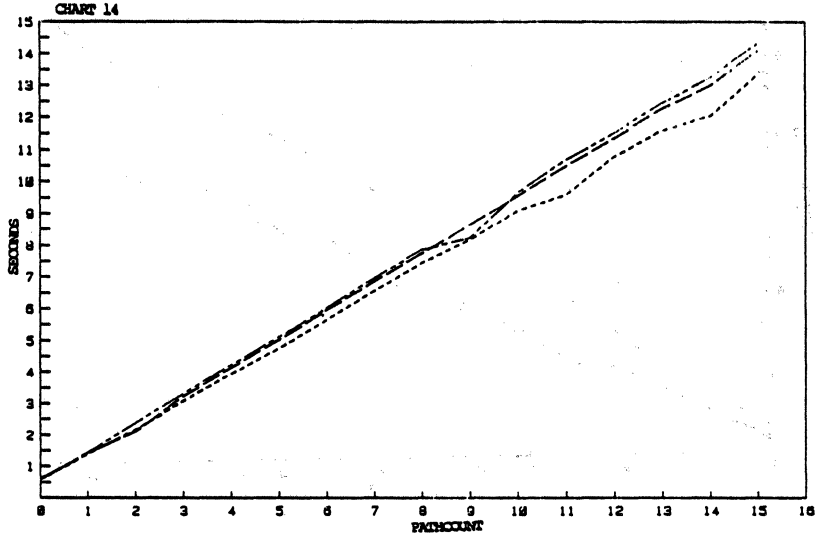
IMAGE-3000 BENCHMARK RESULTS

Time to perform DBPUT's with busy CPU

III WITHOUT LOGGING — — —

44 WITHOUT LOGGING —·—·—

64 WITHOUT LOGGING --------



IMAGE/3000 - BENCHMARK RESULTS

CPU TIME VS. ELAPSED MIN

	SERIES 33		SERIES III		SERIES 44		SERIES 64	
	CPU	MIN	CPU	MIN	CPU	MIN	CPU	MIN
DUMYLOAD=0			770	55	523	48	317	49
DUMYLOAD-10	3784	124	1456	74	1032	62	592	64
DUMYLOAD=20			1705	80	1204	68	679	69
DUMYLOAD=40			2221	95	1576	78	871	79
DUMYLOAD=60			2739	109	1966	92	1052	93
DUMYLOAD/DUMY2=20 NO SPREAD			2106	144	1155	98	651	89
DUMYLOAD/DUMY2=20 SPREAD			1718	82	1279	92	700	81
CPULOAD=1			24722	425	24953	434	23600	402

TABLE 1

REFERENCES

[1] Hewlett-Packard, "IMAGE Reference Manual", page 4-1, August 1981.

[2] ibid, page 104

[3] ibid, page 2-17

[4] Freeman, Peter, "Software Systems Principles A Survey", page 272, Science Research Associates Inc. 1975

[5] ibid, page 275

[6] Montreal Swap tape 1983

[7] Montreal Swap tape 1983

[8] Montreal Swap tape 1983

[9] Montreal Swap tape 1983

[10] HPIUG Contributed library

[11] HOWMESSY, Robelle Consulting
27597-32B Ave.
Aldergrove
British Columbia VOX 1A0
Canada

[12] Hewlett-Packard, "Performance Guide"