Fall 1979

Computer Museum

Journal of the HP General Systems Users Group

Publication 'Bugs' and 'Fixes'

As the membership expands, so must our service. In this effort, the Publications Committee and Manager have established a new article review system and publications deadline schedule. As with any new system, the 'bugs' must be identified and fixed. We are aware that we have extended the publications schedule with this issue; however, the 'bugs' are being solved as we move ahead to improve and expand the material published in the JOURNAL. We believe that the results will justify your patience and cooperation.

Please note that beginning with this issue your JOURNAL will be designated by the appropriate quarter and year, as we reestablish our publication schedule.

If you have any suggestions or would be interested in joining the Publications Committee, please contact the Executive Office. The JOURNAL and NEWSLETTER are your publications. Help us improve them!

Thank you!

John R. Ray, Chairman
Publications Committee

Contributions: HPGSUG Executive Offices
ATTN: Carol W. Taylor
Publications Manager

SPOTLIGHT ARTICLE

Experiences with the Manufacturing Package MFG/3000

by Ivan Rosenberg
National Computer Corp.
San Luis Obispo, California

Abstract

Since June, 1978, HP's new manufacturing package, MFG/3000, has been operational at Vetter Corporation, a manufacturer of motorcycle accessories with plants in California and Illinois. There are three modules to the package: Engineering Data Control (EDC), for parts description, bill-of-materials, workcenter descriptions, and routing; Inventory and Order System (IOS), for inventory control, purchase and work orders; and Material Requirements Planning (MRP), for materials planning. The functions included and missing from these modules, with emphasis on their practical application, are discussed. Operational aspects and recommendations for installation procedures, particularly in a multi-plant organization, are described. An order processing system, from order entry through accounts receivable, has been implemented, revealing some strengths and limitations of MFG/3000 relative to customization. Finally, some potential improvements, such as may result from the new IMAGE/3000, are mentioned.

I. Introduction

MFG/3000 is a collection of three software modules that form an integrated MRP-based manufacturing support package. It is oriented to manufacturers who manufacture and assemble discrete parts, where a primary goal is to minimize inventory investment, yet maintain adequate and timely supplies for production and customer orders. On-line use is encouraged through the use of highly simple menu and data entry screens on CRT terminals, although batch input is available.

MFG/3000 is structured on IMAGE/3000 data bases, and permits the use of QUERY for ad hoc and customized reports, as well as some emergency data base "fixing." The on-line programs utilize DEL/3000, thus requiring HP 264X teminals.

The relation between the three software modules is shown in Figure 1. Engineering Data Control (EDC) maintains descriptive, cost, and planning information about all parts, and bill-of-material, workcenter, and routing information about the manufacturing operation. In addition, engineering change information regarding future changes in a bill-of-material and miscellaneous remarks about a part or bill-of-material may be stored. The Inventory and Order Status (IOS) module maintains records of planned and actual inventory issues and receipts. Work and purchase orders are entered directly into the system. Upon request, IOS traces through the relevant bills to determine and allocate all needed parts for a work order (which can refer to only one fabricated part). At this time the user can determine if all needed parts are or will be available. At the proper time, based on specified lead times, pick lists are generated. The input resulting from the actual pick operation creates a reduction in inventory level and a stock activity record. Likewise, receipts create a historical stock activity record. Such records can be accessed on-line and are purged on a periodic basis.

Finally, the Material Requirements Planning (MRP) module is a planning tool to help management balance current and anticipated demand for a part with current and anticipated supply for the same part.

HP recommends that there be at least two management positions associated with the implementation and operation of MFG/3000. The User Trainer is responsible for educating users as to the proper procedures for using the system, including on-line transactions, management policies, and use of printed reports. The System Administrator is responsible for the proper operation of MFG/3000 itself, including data base integrity and security, maintenance, back-up, and modifications as required.

This paper will discuss the experiences of one of the earliest users of MFG/3000, first from a user point of view (Section III), and then from a System Administrator view point (Section IV). Perceptions gained during implementation of an order processing system are shared in Section V. Enhancements, potential and recommended, are discussed in Section VI.

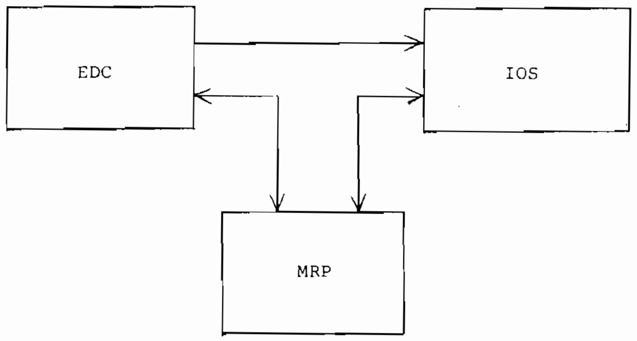


FIGURE 1. MFG/3000

II. System Installation

Vetter Corporation is a manufacturer of motorcycle accessories and employs approximately 250 people at an Illinois facility and approximately 125 at a west coast California plant. It has experienced a high growth rate over the last few years and has had a continuing problem relative to materials required for the production and support of its products. In order to solve these problems, a program of investigating computer systems for inventory management was begun during the summer of 1977.

In the fall of that year, National Computer Corporation (NCC), a computer consulting firm, was engaged to perform a market survey and analysis and to make recommendations regarding feasible computer systems. Among the criteria for the system were:

1. Reasonably powerful data base management system software
2. On-line query capability into the data base
3. Ability to handle multiple data bases
4. Ability to handle up to 24 on-line CRT terminals simultaneously, with at least three background batch streams

(Continued)

5. Availability of adequate maintenance and vendor support

6. Availability of a reasonably powerful MRP application package

7. Capacity for future expansion; Primarily because of the on-line data base query capability, there were few feasible systems to consider.

The HP 3000 and MFG/3000 were studied during the first months of 1978, followed by a purchase of an HP 3000 Series II Model 6 on March 22. Besides MFG/3000, this system included 320K of main memory, two 50 MB discs, a 600 LPM line printer, a 1600bpi tape drive, the MPE-II operating system, the IMAGE data base system, QUERY, DEL for termial management, and COBOL. The 3000 arrived on May 2 and was installed on May 10. The industry specialist from the Los Angeles office installed the EDC package on May 12, and the following week on May 19, he installed the IOS package. On June 9, the MRP package was installed. The system has been in continuous operation since that date. During the fall of 1978 an upgrade was made to a Series III with one megabyte of main memory and a 120 MB disc was ordered. In addition, the new operating system MPE-III was installed in September, 1978. Two 9600 band lease lines connect the Illinois plant with the California computer facility.

After system selection, NCC was contracted to perform facilities management and MFG/3000 administration during installation and until system operation stabilized and Vetter personnel could be properly trained. Turnover of system management was completed during the latter part of summer, 1978. Thereafter, NCC designed and supervised the implementation of a comprehensive Order Processing System—including order entry, accounts receivable, warranty picking, shipping and inventory allocation—which is discussed in Section V.

III. MFG/3000 From a User Point of View

The EDC package is somewhat of a misnomer for production description. It could be more accurately described as a definition module, and more specifically, the definitions are all defined in terms of manufacturing or production criteria. It is broken down into three primary divisions:

1. Part definition—Data elements directly tied to a part number and structure definition, which is the relationship between a number of parts which describe a finished product or sub-component.

2. Planning and control function for purchased parts as well as fabricated goods.

3. Routing and work center section to describe work centers and activities which occur at the work centers to produce finished goods and sub-assemblies.

The IOS package handles inventory, including stock locations, quantities on hand, and other essential elements necessary to track and maintain accurate inventory records. The second portion of this relates to orders—purchase orders and their associated vendor information, and work orders, which are essentially in-house purchase orders and associated pick lists and materials requisitions to drive the work order system.

The MRP package is very complete, allowing multi-purpose policies and scheduling techniques. It is a regenerative type in that it is a batch type program which is run once a week. It is essentially driven by the IOS and EDC packages; and if those two have been brought up in a complete manner, the MRP package simply strips data from both and produces a series of reports which control and schedule both in-house work orders and out-of-house purchases.

Implementation from the standpoint of the users is reasonably defined by the structure of the system. The EDC package must be brought up first and item data entered so that it may be transferred in a batch job to IOS in order to allow the warehousing people to begin to work on their inventory counts. We have found that, with the item data set complete and outstanding purchase orders entered on the IOS data base, the warehousing functions can begin by receiving the outstanding purchase orders and by commencing cycle counts which are provided by the system. This will allow purchasing people to begin to become familiar with entering purchase orders on the system which have been initiated by their old manual inventory control system.

Essentially, this allows a small step toward bringing them out of their old system into an automated one and reduces the trauma of grossly exposing them to all its facets at the same time. The warehousing people at this point are simply exposed to receiving goods on the computer and cycle counts. At this point, two parallel lines develop—the warehousing group can do physical inventory and load the counts on the computer and verify the inventory by additional cycle counts to guarantee the accuracy of their data, and the person responsible for developing the EDC structure information can now begin to load the structure data required to define the assemblies and sub-assemblies required in that plant operation. This data is necessary prior to the development of any internal work order situations. At this point, the purchasing people have been exposed to the system and are loading their purchase orders. The receiving department is receiving goods acquired from those purchase orders. Incoming inspection of those goods may be implemented at any point in this cycle, depending on the local needs. At the same time, the warehousing people are becoming familiar with the cycle counts and details of the system necessary to maintain their inventories.

The final step in the process is — with the structure information loaded—a final review is required of the EDC information necessary to define inventory control and purchase part definition. This information relates to lot sizes, lead times, and policies, etc. It is extremely important that they be related to the real situations in this plant and that they be applied with good inventory management goals in mind and good purchase policies. This is important because of the fact that the MRP program uses this item data in calculating inventory purchases and demand. The system simply emulates the buying decisions and the manufacturing decisions which have been attached to their component part. In essence, poor purchasing information will be followed by the computer and implemented just as effectively by the computer as it would by an individual.

If the EDC control portion has been implemented properly and with all due regard for economy and control and the IOS portion implemented with accurate inventories and outstanding purchase orders, the final step is to load a master schedule useable by MRP to create the materials requirement plan for the plant. The key in developing a master schedule is, of course, to weigh the needs of the sales forecast with the inventory and cash goals provided by a finanace group to provide a realistic schedule within the capabilities of the physical plant which the MPR program can implement to drive purchase order and work order requirements.

One final issue is that the operational departments of the company be structured for control and economy with the master schedule with its inputs from sales forecasting and finance driving the MRP, which provides further specific direction for purchasing and production. It is necessary that the operational departments of that plant be structured so that a scheduling department or production inventory control department be intimately tied to the forecasts, etc. The computer and the MFG/3000 package will take the master schedule demand and provide information which may be directly inputted into purchasing and production scheduling. Conflicts in the schedule must be revised due to the inability of purchasing and production to accomplish the goals. On the other hand, the system also provides quantitative feedback to anticipate and measure capacity limitations. These limitations, if pointed out in advance, may, at the option of management, be resolved to meet the master schedule if its requirements are paramount to these defined limitations.

Our feeling after using the package for approximately six months is that it provides extremely useful and flexible tools for purchasing, warehousing, and production to accomplish their objectives. It provides qualitative tools for management to evaluate operational departments in terms of inventory value, inventory turns, back order analysis, shortage analysis, and materials requirements. These may be used by the operational departments to evaluate their own members and by management to evaluate their operating departments. The MFG/3000 package is entirely materials-oriented, and its biggest limitation is simply that it encompasses the materials portion of the manufacturing plant's operational system. It provides only slight support for labor and routing information and provides little or no support for the financial department. The package may be enhanced best by further development of product costing, job costing for the materials area, and the development of a master schedule system to ease the development of master schedules and inventory control. Obviously, to encompass all the elements of the manufacturing plant, an accounts payable package tied to the purchase order portion of the IOS package is desirable; and on the other side of the master schedule module, an order entry and accounts receivable package would greatly round out the whole system. Of course, standard accounting functions, such as general ledger, etc., should be provided to tie the whole system together.

In summary, the package has been found to be extremely easy to use and is easily implemented by the operational departments. The design is relatively simple and straight forward so that it solves a large number of the operational problems of the manufacturing plant without embroiling the departments in embellishments which detract from the objective of this system—which is to increase the efficiency and operational effectiveness of the manufacturing plant.

In many MRP-oriented systems, the goal of increased efficiency is obscured by "bells and whistles," which reduce the effectiveness of the computer-oriented system. The MFG/3000 package, in a simple, straightforward approach, has addressed the materials problems of the manufacturing plant in a very successful manner. We strongly recommend that Hewlett-Packard continue with this approach and encompass the above-mentioned additional areas to provide an inclusive package for the manufacturing plant.

IV. MFG/3000 From The System Administrator Point of View

A. Installation. As discussed in the previous section, installation was a fairly non-traumatic process. Vetter had the advantage of implementing first at the California plant which was in the process of being established and is considerably smaller than the Illinois plant. In addition, since the computer system was on-site, communications problems were not involved. Implementation of each package for the Illinois plant followed California installation by about one month. Both installations pointed out the need for careful planning, particularly during the loading of the EDC data base. For the System Administrator, early establishment of the default values for the various data items associated with parts is very important. Although direct use of QUERY can be used for some simple changes to large numbers of parts, such updates must be done extremely carefully. Later, QUERY was used indirectly for massive updates by using it to create a transaction file containing all needed changes for EDCMAINT (the EDC update program). Such a technique can also be used to more easily request reports pertaining to large numbers of parts, instead of requesting the report for each individual part number through EDC. For example, one can request a bill-of-material for all parts meeting certain criteria. It is recommended that forms similar to the EDC screens be designed to simplify and control initial data entry.

As is typical, most errors are discovered during system use, so one can expect to detect most of the errors in EDC during the early months of IOS use.

Another problem was encountered during the installation of the Illinois plant. Although MFG/3000 is installed by HP assuming a MGR.MFG3000 user and account, with two plants two different accounts are needed (each plant has it own MFG/3000 data base). In addition, to save disc space and CST's we decided that only one copy of MFG/3000 programs would be stored for use by both plants. With all programs stored in one plant's accounts, the other plant needs the work files, its own data base, the forms files, and its own copy of the JCL (or STREAM) files. These files are modified as follows:

1. The PGM=parameter of the EDC3000 and IOS3000 screens (the first screens of the modules) must be changed to a fully qualified program name in the "program" account. This is easily done through FORMAINT.

2. All JOB user and account names must be changed to that of the plant.

3. All RUN statements in JCL and those of on-line users must be changed to fully qualified program names.

4. The access security on the program groups must be changed to ANY for execute. A similar change may have to be made in the account in which the programs reside.

(Continued)

(Spotlight Article Continued)

Adequate security between plants is maintained since file references default to the log-on account, and this also causes the proper data files to be referenced. Access to the data bases is still denied to anyone not logged into the account containing a copy of the data bases, work files, and forms. The MFG/3000 programs simply will abort if such an account attempts to use them.

An advantage is gained since the account which does not contain the programs requires only IA and BA capability to run MFG, although more capability is required if QUERY procedures are to be created and used within the account. We have established a "user" and a "manager" user for each plant account, with the "user" having very limited capability, being the logon for all normal users. The use of the "manager" user is restricted to the System Administrator and the EDP department.

It is necessary to have a hard copy device attached to the terminal to be used for inventory control so that the Material Receiver Reports may be printed. There appears to be no way to redirect these to the system printer or a disc file; thus, if not printed, they are lost.

If possible, perform all part number conversions before installation. Vetter has peformed such a conversion after installation, the techniques and result of which will be reported at some future date.

B. Training and Documentation. Because Vetter was one of the first purchasers of the MFG system, we were also the first to attend the courses offered by HP. There is one course for each of the three modules, each course divided into two sections, one for the System Administrator, and one for the User Trainer. It was our recommendation that these courses be attended only by people who were already educated regarding computers and manufacturing, thus reducing the chance they would degenerate into introductory courses.

We found the quality of the courses, instructors, course materials, and documentation to be very high. Each course lasts two to three days, and much material of practical worth is covered. We had an advantage in that the implementation of each module was actually accomplished before the corresponding course was attended, but that just increased its worth to us since problems we had encoutered and "tuning" issues could be discussed. There was considerable lab work, although some was simply rote. There is a significant advantage in having a "live" terminal in front of each student to test functions as they are discussed.

The system specialists in MFG were also (and continue to be) invaluable, as problems occur particular to our installation. In the beginning, telephone calls averaged one or two a week, and we were visited at least once a month. This frequency diminished considerably after the first half year of use.

C. Customization. Vetter purchased the object code version of MFG. The source code version costs considerably more and is not maintained directly by HP. Despite this, MFG offers some customization abilities.

Screens may be modified under certain conditions. Alternatives may be added to menu screens, including branching to user-written routines. Fields that exist in the standard MFG data base may be added to data entry (FMT) screens and non-key fields may be deleted from those screens. The order of existing fields may not be changed. Fields added to the data base by the user may not be added to the screens.

Data retrieval (RET) screens, for all practical purposes, may not be modified except to move fields (maintaining the same order) and to change protected fields.

The new user may become somewhat confused by MFG's use of the NEXT= parameter of a DEL screen, which usually indicates the next screen in a sequence. In MFG it indicates the previous screen, and is used to "back up."

Field editing may be changed within the different alternatives offered by MFG (such as alphanumeric, numeric, etc.) as long as the new editing is more restrictive. The size of fields may be diminished.

We have modified the JCL streams often, particularly with the MRP module. Up to 15 levels of planning are available; Vetter uses only seven. Since the analysis routines are performed by chained STREAM commands, one need only change the last STREAMs of the the last level desired, e.g., MRP2007J, to chain instead to the last jobs of the MRP process (MRP 2100J and MRP 2400J). In addition, MRP 2400J must be modified to turn the FILE and SORT statements for unused levels into comments. Finally, the FILE statements in the last MERGE are likewise comments. Finally, the FILE statements in the last MERGE are likewise deleted. Thus, to add another level of planning is a relatively simple process of removing comment indicators.

It is difficult, if not impossible, to modify MFG off-line reports. QUERY has proved to be an invaluable tool in this regard. In addition to the technique of producing a transaction file described above, QUERY is extensively used directly for producing a multitude of periodic and ad hoc reports. Purchasers should be cautioned, however, not to permit any except trained technical personnel to use QUERY for updates. It is very easy to get the data base in such a state that the pointers, etc., are in very bad shape. Additionally, updates lock the data base for a long time. In general, avoid using QUERY for anything except retrieval.

D. Security. Probably one of the weakest aspects of MFG is its security provisions. There are essentially four levels of security: the MPE account structure, file lockwords, MFG originator numbers, and data base passwords.

The MPE password security is of limited value since it must be known by large numbers of plant personnel to enable them to use the system. Lockwords on user programs suffer from the same problem, although all our System Administrator routines are so protected.

Originator numbers must be entered by the MFG user before access to most on-line functions. Some data retrieval and report requests do not require originator numbers. These numbers, ranging from 0 to 99, are assigned to individuals and groups and are used for distributing the Transaction Register Report of EDCMAINT and for controlling the functions that may be performed using each screen. An originator may have any of three capability levels:

1. None
2. Modify
3. Add/Delete (which includes Modify)

Although access rights are assigned through MFG by fields, we have found it more practical to regard the capabilities to be attached to screens since to effectively use a screen, one must have the same capability for all fields on that screen. Any particular originator must have Add/Delete capability for all fields associated with him, not just a subset.

Originator numbers do have some limited value in protecting against inadvertent but unauthorized behavior, but they have almost no value in protecting against intentional misuse. Originator numbers are restricted to only 100 alternatives and are printed on so many reports that acquiring one seems a very easy process. We have recommended to HP that the originator number be alphanumeric. Although we were told that we could have 100 originators as long as we had only 20 variations of capabilities, we discovered that only 20 originators are allowed, regardless of capabilities.

The system is installed with a system administrator originator number with complete capability (in fact, there are numerous originators existing on the originally installed system which might be deleted). It may be best not to have any originator with such capability, and only create it when needed. At the very least, the standard administrator number should be changed since it is published in HP documentation.

A major criticism of the security is the data base protection. Any user with the logon password and the data base password can easily access (and update) the data base using QUERY. Unfortunately, there is one master password to write access to all fields in the data base. This password is common to all three MFG data bases, is the same for all MFG installations, is not at all cryptic, and even worse is published in some HP literature! For the purchaser with dial-up capability, a major vulnerability exists with regard to the security of the data base. We have recommended to HP that a unique password be established for each installation.

E. Production Operation There are many batch jobs that must be STREAMed to maintain the system. Data entry in EDC does not directly update the data base, but rather appends a record to a transaction file which is later used by batch job, EDCMAINT, to perform the updating. EDCMAINT also produces the off-line reports. I0S0400J strips certain information from the EDC data base and updates part information in the IOS data base. Other batch jobs update the Edit Tables (which define screen field editing and originators), delete parts from the IOS data base, etc. In addition, we have created job streams for some QUERY reports.

Most of these batch streams require exclusive access to the data base for correct operation. EDCMAINT is particularly tricky, since it repeatedly requests and gives up exclusive access. If someone logs into EDC between such accesses, EDCMAINT can abort in the middle. It is not a big problem to restart it in the aborted place, but it must be done with care and is an inconvenience. In general, all batch jobs are run after working hours and/or during lunch. There is a slight modification to the EDCMAINT jobstream that will prevent logons during its execution; but this poses some additional problems if an abort occurs for other reasons. And, since the MFG programs are shared by both data bases, this prevents users of the other data base from running EDC. One then tries to insure that all users of MFG are logged off prior to STREAMing EDCMAINT. A potential solution is to create two users, e.g., USREDC and USRIOS, one for each module. Another simpler alternative is to ask users to append their name and module in the HELLO command, e.g., :HELLO IVANEDC, MGR.MFG3000. A :SHOWJOB then indicates who is logged on. Without this additional information, one only has the QUIET indication on the SHOWJOB as a guide.

It was originally intended that EDCMAINT would run only once a day. However, during the first few months after installation, changes were so frequent that four runs in a day was not uncommon. With more users on now and a more stable data base, there are two EDCMAINT runs per day, at Noon and after 5:00 p.m., for each plant. There is a time zone difference between the two plants. Thus, because of the extensive nature of EDCMAINT, users at one plant can experience a noticeable degradation in response time when EDCMAINT is being run for the other plant.

To reduce such degradation, one may remove from EDCMAINT the job steps which update data sets for which the System Administrator is certain no transactions exist. For example, Vetter does not yet use the workcenter or routing data sets. Clearly, such a modification must be done with care.

In addition, STREAMing I0S0400J (which updates the IOS part information from EDC) has been appended to the end of the EDCMAINT jobstream so the IOS data base is always current with the EDC data base. All scheduled off-line reports are then run after the completion of EDCMAINT.

It is very important that the System Administrator monitor the EDCMAINT runs since we experienced frequent aborts and erroneous data entries during the first months of operation. The aborts were not due to program bugs, but rather to something wrong with the input data or data base. The JCL comments and abort messages are very extensive and helpful. However, it is important to insure that EDCMAINT runs to completion before permitting users to initiate EDC again. The Transaction Registers should be distributed promptly to the originators after being reviewed by the System Administrator for serious or frequent errors. We tried to insure that all users completely understood how to interpret the Transaction Register Report and that they kept all copies on file.

In particular, during the initial months of use, the System Administrator should monitor the sizes of the data sets of each data base. This may be easily accomplished using the DBSTATS routine. Capacities should be adjusted so the data sets are approximately 70% full (or less).

Periodic DBUNLOAD and DBLOAD of a data base can also contribute to improved response times. In a multiple disc installation, it is preferable to place the root file on one disc and the data sets on the other, thus reducing disc contention.

F. Multi-Plant Operation. The modifications necessary to the JCL, etc., to maintain only one copy of the programs in a multi-plant environment has been discussed in Section IV.A. Three problems remain for the remote plant:

1. To direct printed output to the remote line printer

2. To be able to defer the printing of certain reports for the loading of special forms, for large reports, etc.

3. To be able to view the JCL listings resulting from a

(Continued)

STREAMed job to check on the occurrence and reason for an abort, etc.

Initially, we installed a "minispooler" provided informally by HP. This routine solved the first problem, but not the other two. Understandably, frustration was high at the remote site.

In the fall of 1978, the minispooler was replaced with the RSPOOL Package of DataCon of Oregon, resulting in the solution of all three problems indicated above.

The JCL for the remote site's MFG system must be modified in the following manner:

1. Add the OUTCLASS = LP,1,1 clause to all JOB statements. This specifies 1 copy and an OUTPRI of 1.

2. After the FILE statement for the report file, a set of about 10 lines must be added to initiate and control the execution of the routine SPOOLCOM. This connects the report files to the remote printer, specifies the number of copies, and permits the report file to be held after printing.

Finally, the program RSPOOL must be initiated. This sets parameters as lines/form, number of lines between forms, etc., and controls the printing of spooled reports on the remote printer (as submitted by SPOOLCOM).

If the OUTFENCE of the system is set to 1 or greater, the JCL of the remote plant will not print on the system printer, but will be held in the spool files. The remote plant then uses SPOOK to look at the JCL files for a solution to the third problem indicated above. Since SPOOK permits access only to the JCL of the logon account, such users do not have access to the JCL of other accounts. SPOOK permits the user to list the job numbers of the JCL currently in the spool file, which then may be used to display the JCL of the job. The users must periodically purge the JCL files in the spooler so that it does not fill.

If the parameters of the RSPOOL and SPOOLCOM executions are set appropriately, the printing of a report at the remote site may be deferred. SPOOLCOM may then be used on-line to alter file specifications in order to initiate printing when ready or to delete a report file.

We have experienced great satisfaction with this arrangement. Essentially, it grants to the remote site all the power (and then some) of the on-site installation.

G. Backup. One of the most important jobs of the System Administrator is scheduling and supervising backup procedures.

Vetter does a partial backup each night, with a full system backup once a week. Currently, the MFG data bases are not separately backed up using the DBSTORE program, although this will probably be implemented shortly since restoring from the SYSDUMP tape is less reliable and slower.

The transaction file of EDC provides some roll-forward capability, since the last five transaction files are automatically saved. However, this requires that a copy of the data base corresponding to its state prior to the EDCMAINT run be available. If EDCMAINT is run twice a day and backup is performed only once, roll forward can be done from only two or three backup copies, not five. However, it is possible to modify the EDCMAINT JCL so that more than five transaction files are saved.

IOS automatically provides a journaling of all transactions which affect the data base. The documentation is very unclear as to how to control this capability. In summary, the logging in IOS is always running. The only control the user has is to which device the logging is directed, disc or tape. Normally, we log to disc. Twice a day, STARTLOG is run to dump the current contents of the disc log file to tape, which also directs any future log transactions to the tape. After the disc file has been dumped, STOPLOG is run to direct future logging to the disc. We insure that no one is logged into IOS during the time STOPLOG is run. Besides insuring that the log file is large enough to contain the number of transactions likely to be entered between tape dumps, that is all there is to it. There is a utility to display the current number of log entries in the file so that it can be monitored. Our volume is easily accomodated on a 1200' reel.

At the current time, IMAGE does not provide any journaling capability; thus roll-forward and roll-backward are not available outside of the MFG facilities.

Another backup capability may be provided by the HP 2645A terminals themselves if they have the tape option installed. If the computer system is unavailable, or data entry takes place over a slow speed data line, the data may be loaded off-line through the terminal onto a tape cassette, then dumped to MFG in a rapid fashion. In order to prepare for this process, store the MFG screens on a tape cassetts, (by displaying a screen, then in local mode copying to tape). Later, in LOCAL mode, display the selected MFG data entry screen by copying from tape to the screen. Put the terminal in formate mode using CTRL f_4. Enter the data normally. Pressing the ENTER key will copy the unprotected fields to the tape and clear the screen. Data entry may then continue in a similar fashion. Conclude by releasing format mode using CTRL f_5.

In order to enter the stored data, log on to MFG and display the appropriate data entry screen. Copying a record from tape to the screen and then pressing ENTER will cause MFG to read the screen in the normal manner.

V. Implementation of an Order Processing System

NCC and Vetter have implemented a comprehensive order processing system (OPS) to interface with MFG/3000. The overall structure of the system is shown in Figure 2. It consists of the following elements:

1. Part Element—To maintain descriptive and availability information about each part that may be sold.

2. Dealer Element—To maintain information about each customer.

3. Order Entry Element—For the entry, review, modification and reporting of information about orders.

4. Pick List Element—To produce pick lists on a selected basis.

5. Shipping Element—To produce invoices and other shipping documents.

6. Accounts Receivable Element—To maintain accounts receivable information relative to orders and dealers.

7. Warranty Element—To maintain information about the product and the end purchasers.

The Part Element operates much like IOS in that it strips relevant information from the EDC and IOS data bases and puts it into the OPS data base. This continues the

MFG policy of maintaining data base separation between major functions. Because IMAGE permits locking only at the data base level (see Section VI), separate data bases permit an acceptable response time in a multiple user, on-line updating environment at a cost of duplication of data and increased storage requirements.

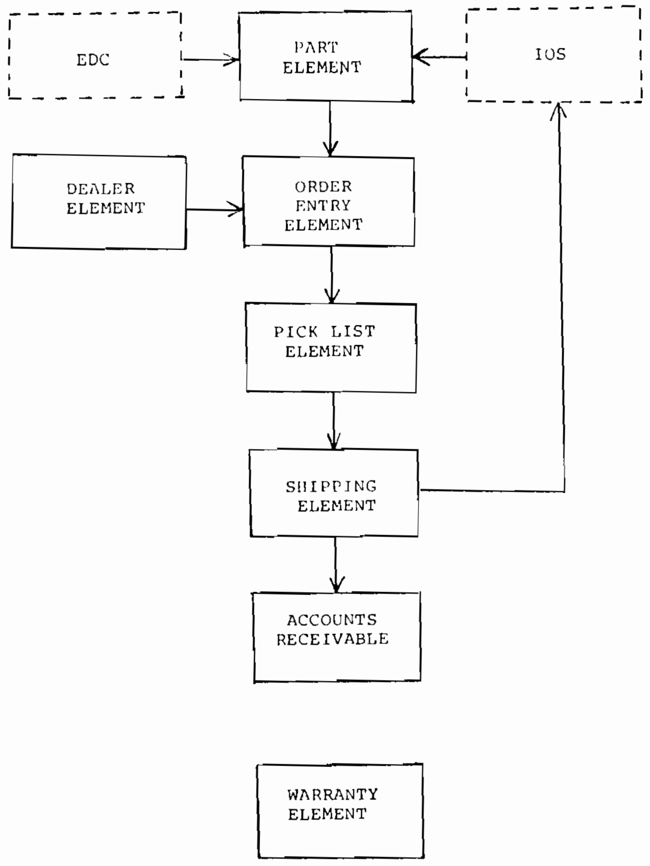


FIGURE 2. THE ORDER PROCESSING SYSTEM (OPS)

In addition to the obvious on-line functions, there are numerous reports (well over 20) provided.

The major contribution of this system is timely indications of the ship date of an order through consideration of planned receipts of finished goods and shipping rate limitations. We expect to be able to predict ship dates within a day's accuracy at the time of order acceptance. Information concerning planned receipts is acquired directly from the work orders and purchase orders of IOS. Thus, any change in production schedules will be immediately reflected in changed order ship dates.

During this implementation, we realized that although MFG and IMAGE will permit the user to add data fields to the end of data sets, this was a dangerous practice. Future versions of MFG could very well expand the number and size of data fields. The addition of custom data fields could prevent the easy updating of MFG, requiring special programs to unload and load the data base (since the new HP fields would have to be "inserted" between the standard and the custom fields), and the custom programs would also have to be changed. This was another major reason that we decided to implement a separate OPS data base.

There are four interfaces between the two systems:

1. Moving part and inventory information from MFG to OPS
2. Acquiring the master production schedule (MPS)
3. Acquiring the shop calendar
4. Updating the inventory counts as a result of shipments

Acquiring part information is a very easy task. The Part Element has produced the side benefit of insuring that the EDC data bases of the two plants contain the same information about the same part number.

The MPS appeared relatively difficult to acquire directly from IOS. We considered having each plant manager create an EDITOR file, from which we would generate appropriate transactions to IOS for MRP and OPS. However, this seemed too complicated. We decided to simply scan the IOS data base for work and purchase orders for finished goods and create corresponding allocation data records in the OPS data base. Since the factory works directly from these work orders, we are using the best information available. This frees the plant manager to use the input of an MPS to IOS as a planning and "what if" tool. Finally, it encourages the plant managers to plan their production schedule out to the horizon of shipment planning.

MRP generates a shop calendar which we use directly for assigning order ship dates. In this manner, we can easily take into account holidays and weekends.

Finally, since IOS provides no method (other than stock adjustments) for drawing down finished goods inventory, the OPS system must create a transaction file of stock adjustment requests that an MFG batch job uses to update the inventory levels in IOS. In addition, OPS generates an Issued Goods Report for cross-checking IOS stock activity records.

All of these interfaces have been relatively simple and easy to implement. It appears that MFG lends itself very well to supporting user-written custom programs. The required MFG documentation is very adequate and clear.

We have experienced two problems with the use of DEL in this implementation. Since DEL does not permit field-level reads and writes, the entire contents of the unprotected fields must be transmitted, even if only a subset of the fields is desired. In a remote terminal environment, this tends to radically increase communications line load, and if the terminal is slow, response time can suffer. Our only current solution is careful design of screens.

The other limitation appears to be in FORMAINT. The OPS employs a few screens where there are a considerable number of enhanced, unprotected fields. We have discovered that FORMAINT (and DEL) will accept a maximum of 1920 characters, including the ESCAPE sequences for the fields. Since there may be as many as 10 characters associated with a field just for the enhancements, this limit can rapidly be reached. Although the 2645A terminal may support a form, FORMAINT may not. Our only current solution is to give in and reduce the number of fields.

To reduce on-line response time when a variety of forms are used, the package FLIPPER by Systech was incorporated into OPS. This permits a number of screens to be stored in the terminal's memory. Thus forms changes may take place as fast as the NEXT PAGE function.

OPS has now been running in a production mode since

(Continued)

(Spotlight Article Continued)

January 1979. It's highly modular design and structured code have lead to low maintenance requirements, easy modification, and rapid enhancement.

VI. Potential and Recommended Enhancements

Enhancements from the user point of view have been discussed in Section III. This section will focus on improvements from a System Administrator view.

With the announcement of the new IMAGE with record-level locking, we had hoped that the three MFG date bases would be merged into one. This would produce significant improvements in storage requirements and some improvement in user response time (since interactive programs will no longer be locking the entire data base). Some operational problems (such as exclusive access aborts) would be reduced. The batch programs that move information from one data base to another would be eliminated, and EDC could be converted to an on-line updating system. However, this is not to be. The limit on the number of item names in an IMAGE data base is a major obstacle. In addition, the development of MMS as a potential replacement for MFG probably precludes extensive MFG changes at this time.

Because of the problems associated with MFG updating, we plan to maintain a separate OPS data base, but record-level locking will produce very significant operating improvements in order processing. The impact of VIEW/3000 as a substitute for DEL could be considerable as the ability to do field level read/write would reduce communication loads considerably. We would also like the size limit on forms to be at least increased, if not eliminated. A signficant improvement to DEL would permit the selective modification of a form, without re-entering all the edit specifications. In addition, it would be nice to simply state that no edit specs existed for the entire form, thus skipping laborious entering of X's. All these problems appear to be addressed in VIEW, which we believe to be used in MMS.

Although MFG provides some journaling, we would like to see the logging from IOS and EDC better coordinated and similar. Rollback abilities would be nice. We are looking forward to have such recovery facilities available in the next IMAGE release so user-written programs could take advantage of them.

The OPS would benefit by a better method of entering the master production schedule into MFG than as a series of work orders.

However, the two major limitations of MFG/3000 are its security provisions and the high number of CSTs it demands. The security provisions were discussed in Section IV. Despite the large size of the Vetter system, it frequently runs out of CSTs, particularly during a COBOL compile. This is because both COBOL and MFG make high demands on this limited resource. In a multi-plant environment, we have doubts that we could operate if a copy of the MFG programs were required for each plant. This limitation has forced us to violate the generally accepted HP3000 practice of small routines in the implementation of the OPS, with its attendant problems of long compiles and decreased system performance. We expect the MFG's high use of CSTs will be alleviated in future updates.

VII. Conclusion

Both as user and system administrator, we have found MFG/3000 to be a well-designed, reliable, well-documented, and "friendly" system, with a few relatively minor limitations. Installation proceeded remarkably smoothly and rapidly. Users became quickly familiar with the system. The addition of custom systems that interface with MFG is straightforward. And, lest we forget, it also has resulted in improved plant operation and customer service.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the assistence of many HP and Vetter personnel who have contributed their creativity and knowledge to the running of MFG/3000 and the ideas in this paper. In particular, we thank Margie Coolidge, DP Manager, and Mark Laslo, Operator, both of Vetter Corporation in California, and John Scribner and Dave Hesser of Hewlett-Packard.

FEATURE ARTICLE

SPL SUPPORTS IMPLEMENTATION OF DIJKSTRA DO-OD

by: **Jung Pyo Hong**
University of California
Los Alamos, New Mexico

ABSTRACT

An implementation of Dijkstra do-od is presented. The minimum requirements to do so for the languages that have if-then-else and do-until contructs are discussed. Examples of programs that have executed are presented showing the compactness of programs written with this form. Systems Programming Language (SPL) supports the do-od construct with no changes to the compiler.

INTRODUCTION

Dijkstra[1] derives a construct for repetition from the notion of guarded commands. This contruct will be referred to as the do-od construct and has the form

(1) do $B_1 \longrightarrow S_1 \square B_2 \longrightarrow S_2 \square \ldots \square B_n \longrightarrow S_n$ od

B_1 is a Boolean expression and S_1 is a statement sequence. S_1 is said to be guarded by B_1, and box is used as a delimiter. The meaning of (1) is:

(a) A true B_i permits the execution of its statement sequence S_i.

(b) The do-od statement is finished when all B_i are false. Otherwise (a) is repeated.

The B_i-S_i pairs are not ordered. The user of the do-od does not care which B_i is tested first nor if a subset of B's is treated simultaneously; the user cares only that the do-od terminates with every B_i false. This end condition is useful in correctness agruments.

Interestingly, (1) introduces notions of concurrency. All guards can be tested and the appropriate statements executed until all guards are false. Or, one can think of each B_i-S_i pair residing in a separate machine.

The indeterminacy implied by (1) is not mandatory in an implementation. Sequential testing of the guards, for example, is permitted.

IMPLEMENTATION

This implementation relies on a language already supporting the if-then-else and the do-until statements. Access to the top of a stack is required to allow nesting of the do-od.

The method employed simply replaces $\longrightarrow$, $\square$, do, and od with key words of the language. The mechanism for the substitution may be a prepass or simply an in-line expansion of symbols as the source code lines are processed. First, $\longrightarrow$ and $\square$ in the construct are replaced with symbols better handled by commercial equipment, $\longrightarrow$ by iterate and $\square$ by ordo so that (1) becomes.

(2) ddo B_1 iterate S_1 ordo . . . B_n iterate S_n odd.

Access to a stack will be donated by tos. Its use on the left of := means that an expression value is pushed onto the stack. Its use elsewhere means that a value is taken from the stack.

Implementation is accomplished using Systems Programming Language (SPL), designed in the early 1970s, supported on the HP3000 computer. Its syntax, pertinent to this implementation of the Dijkstra do-od, is listed in (3). If-then-else has the usual meaning; so does do-until. Define is SPL's implementation of a macro. The symbol getter (upon recognition of a defined symbol) expands the symbol and places the expansion in the input stream and processes it, extracting the next symbol. The syntax of if-then-else and do-until in the language that supports this implementation is in the form suggested by N. Wirth,[2] with deletion of annoying quotation marks. The braces, {}, are used to mean repetition of their contents (zero, one, or as many as is desired).

(3)

if-then-else	= if condition-clause then statement else statement.
do-until	= do statement until condition-clause.
condition-clause	= Boolean-expression.
statement	= statement; Begin {statement {; statement}} end.
define-statement	= define {identifier=string# {,identifier=string#}}.

The required substitutions for the implementation of do-od (2) are:

(4) define

ddo	= do begin tos:=false; if#,
iterate	= then begin#,
ordo	= end else if#,
odd	= end else tos:=not tos end until tos#

In this implementation ddo odd, a null statement, is caught at compile time as an error.

It can be seen that expansion of (5) meets the syntax requirements of the language (3).

(5) ddo x>y iterate x:=x-y ordo y>x iterate y:=y-x odd;

EXAMPLES

The implementation was motivated by the desire to run some of the programs given in Ref. 1 on a computer. An example from p. 45 of Ref. 1 is: Find the greatest common divisor of two positive numbers using Euclid's algorithm.

```
x:=45; y:=95

ddo x>y iterate x:=x-y

ordo y>x iterate y:=y-x

odd;

output (x);
```

Another example comes from page 49 of Ref. 1: Operating on four variables, output the greatest common divisor and the least common multiple of x>0 and y>0. *(Continued)*

```
x:=45; y:=95; u:=95; v:=45

ddo x> y iterate x:=x-y; v:=v+u

ordo y> x iterate y:=y-x; u:=u+v

odd;

output ((x+y)/2); output ((u+v)/2);
```

CONCLUSION

If one begins with an ALGOL - or Pascal-like language that supports access to the top of a stack, then an implementation of Dijkstra ddo-odd can be obtained by a simple source-symbol expansion. The SPL compiler supports Dijkstra do-od with no changes to the compiler. Programs written in this notation show remarkable clarity and conciseness.

REFERENCES

E. W. Dijkstra, A Discipline of Programming (Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976).

N. Wirth, "What Can We Do About the Unnecessary Diversity of Notations for Syntactic Definitions?", Federal Institute of Technology (ETH), Zurich, Switzerland (1977).

SPL/3000: OVERVIEW AND COMMON ERRORS

by: Robert M. Green, President
Robelle Consulting Ltd.
Delta, B.C., Canada

The HP 3000 does not have a normal assembler language. Instead, it has SPL, a high-level, machine-dependent systems programming language that is based on ALGOL. All systems support software for the HP 3000 is written in SPL, much of it without recourse to assembler-level codings.

When implementing large on-line commercial applications for the HP 3000, it is useful to be able to use SPL. Without access to a small amount of SPL coding, it is difficult to achieve significant optimizing results.

ADVANTAGES OF SPL:

1. Well-structured programs can be written because of the rich array of control structures in SPL:

   IF-THEN-ELSE.
   DO-UNTIL.
   WHILE-DO.
   CASE-OF.
   BEGIN-END.
   PROCEDURES
      PARAMETERS
      LOCAL STORAGE
      GLOBAL REFERENCES
      LOGICAL FUNCTIONS

2. The SPL compiler generates very efficient and compact code, and does not generate overhead data structures. As a result, stack and code segments are as small as possible.

3. SPL handles character manipulation efficiently and flexibly. You can create a powerful data editing/formatting library customized to your needs.

4. The SPL compiler is very fast compared to the COBOL compiler.

5. With SPL there is no run-time interface between your program and the terminal user. You can design a much more "friendly" program than by using the COBOL ACCEPT verb, since your program examines every user key stroke.

6. Debugging of SPL programs is straighforward, using MAP, PMAP and DEBUG. Breakpoints can be placed at any line of source code (unlike COBOL).

7. SPL programs can call all system intrinsics directly, while COBOL programs cannot.

DISADVANTAGES OF SPL

1. SPL data structures cannot be as complex as COBOL (i.e., there are only one-dimensional arrays of simple variables). However, the features of COBOL can be simulated using EQUATE, DEFINE, equivalencing and naming conventions. With DSETSPL (a utility program marketed by Robelle), source code required for IMAGE/3000 (buffers, set names, field lists, keynames, etc.) is created automatically from a database itself.

2. SPL data structures do not have a fixed size as in COBOL. You must specify the size in each statement. This gives you added flexibility, but, on balance, is a disadvantage. By using names for such constants (EQUATES), you can arrange the program maintenance so that a single change to the constant declaration will be reflected in all statements.

3. With the power of SPL, it is tempting to indulge in machine-level trickiness. Programmers must be well-disciplined to avoid ASSEMBLE/TOS and other features of SPL that can obsure the meaning of a program.

4. All input/output must be done by calling system intrinsics. SPL does not have an INPUT or OUTPUT statement. This can be turned to advantage by designing custom terminal interfaces. Access to databases is identical to COBOL, but is more efficient, since commonly required data can be global and need not be repeated in each "subprogram".

5. Neither decimal arithmetic (COMP-3) nor multi-word binary (COMP, with implied decimal) is directly supported by the compiler. SPL/AIDS (available from Robelle) contains SPL source code for using packed decimal, as well as for solving other common problems.

6. SPL does not have any feature comparable to the COBOL "copylib". Maintenance of programs is more difficult when record structures are changed, since every source file must be modified. With QEDIT (the program development system from Robelle), $INCLUDE commands are interpreted at compile time to merge standard source with your program.

7. Experienced SPL programmers are seldom available on the job market, necessitating training for all new employees. However, if programs are going to consist mainly of calls to standard HP routines (IMAGE, VIEW, etc.), a considerable amount of training is required anyway. In these cases, the resulting COBOL program is certainly not portable and might just as easily have been written in SPL. Suitable SPL training can usually

be arranged from other experienced users and consultants.

SUMMARY

Use SPL for system utilities (ex: file copier), for commonly called library routines (ex: edit checks), and for high-volume data entry applications. Use COBOL for batch reports and low-usage on-line applications.

Twenty Common Errors in SPL/3000

1. "MISSING SEMICOLON"

If you forget the semicolon at the end of a statement or declaration, you get this error at the end of the next symbol:

```
INTEGER A
A:=1;
^
MISSING SEMICOLON
```

Every statement must have a semicolon at the end of it to separate it from the next statement, unless the statement is followed by an END (then the semicolon is optional). The only time a statement should not have a semicolon after it is when the next symbol is an ELSE or UNTIL.

2. "ILLEGAL STATEMENT BEGINNER"

The most common cause for this error message is a previous error in the program. For example, any error in a declaration will generate this message on the next declaration; it also results when you forget to declare a variable or procedure, or if you misspell a word, or if you miss a BEGIN or END (see #5).

3. "MISSING COLON"

This error occurs when the compiler sees an unknown symbol (never declared or misspelled) and decides that it must be a LABEL (since they are the only symbols in SPL that do not need to be declared). Since LABELs are followed by a colon (:), you get the MISSING COLON message:

```
INTEGER TRIANGLE;
TIANGLE:=1;
         ^
         MISSING COLON
```

4. "MISSING ASSIGNMENT OPERATOR"

SPL uses a colon-equals (:=) as the assignment, not an equals sign (=), as in most other high-level languages:

```
INTEGER A,B,C;
A=B+C;
 ^
 MISSING ASSIGNMENT OPERATOR
```

5. Unmatched BEGIN and END

Many strange and puzzling errors are generated as the result of mismatching BEGIN and END pairs. When you have too many BEGINs, you will get error messages from the compilers at structure points ("UNTIL", "ELSE", etc.). The most common symptom of a missing END is the message ILLEGAL STATEMENT BEGINNER on the next PROCEDURE declaration; nested procedures are not allowed. When the compiler reaches the "END." that ends your source file, if the BEGIN and ENDs do not pair up exactly, the message "BEGIN END DO NOT MATCH" is printed. The SPL compiler prints a BEGIN-END counter on the program listing (third column, 1 digit) to aid in detecting these errors. The counter is incremented on each BEGIN and decremented on each END. At the start of a new procedure, the counter should always be "1".

When you have too many ENDs (or forget a BEGIN), the results are less spectacular, but equally damaging. Since the compiler stops compiling when the BEGIN-END counter decrements to "0", it usually compiles your program without any error messages (but skips the lines after the BEGIN-END counter reaches "0", including the mainline). If you do not set a listing when you compile or prepare, the only symptoms of this problem occur when you run the program. Since there is no mainline (outer block, OB' or MAIN=XXX), the program immediately terminates. Another puzzling error occurs when you forget (or misplace) the initial BEGIN, because the compiler skips over all source lines until the first BEGIN.

6. Procedure Problems

Although a procedure can be thought of as a small program, there are certain differences between declarations and statements that are local and those that are global. You cannot initialize local arrays (simple variables are okay), unless you declare them as "=PB" (but these arrays are for constant values only, and cannot be indexed if type BYTE or DOUBLE). Local variables are dynamic (global are static), which means they do not retain their value between calls. In order to retain a value in a local variable, declare it as OWN. Nested procedure declarations are not allowed; that is, a procedure cannot be declared inside another procedure, unless it is merely an external reference. For subdivisions of code within a procedure, subroutines are allowed, but these must be used very carefully (see #17).

7. GOTO No-No's

You should never GOTO into (or out of) a FOR loop or into a SUBROUTINE. GOTO's are seldom a good idea in SPL, when there are so many other structures available.

8. "EXPECTS CONSTANT"

There can only be one array initialized per declaration:

```
ARRAY A(0:5):=6(0),B(0:5):=6(0);
                   ^
                   EXPECTS CONSTANT
```

9. Zeroeth versus First

In SPL, everything implicitly starts at 0, not at 1, as in most other high-level languages. For example, array names without subscripts refer to element 0, CASE statement entries are numbered from 0, and array elements passed to procedures are treated within the procedure as establishing the zero element of the array parameter.

Examples:

```
INTEGER ARRAY A(1:10);
READ(A,10); <<means READ (A(0),10),
not READ (A(1), 10)>>
IF A="  " THEN ... <<means IF A(0)="  "THEN ...>>
```

(Continued)

Although you can declare any array with any lower bound and upper bound, it is best to set the lower bound to 0. When you refer to the array name alone, you are referring to the "zeroeth" element of the array (even if there isn't one), not the "first" element. Since SPL performs no bounds checking on array references, this can lead to puzzling errors. (Another common mistake is to think that the array name by itself refers to the entire array. If you have two arrays (A and B), "IF A=B THEN" means "IF A(0)=B(0) THEN". SPL does not treat arrays as independent entities, only as places in memory followed by some space. Also, remember that only BYTE array can be compared for more than one element, and then only if you specify a byte count or a literal constant.)

You should also be aware that most things in MPE also start at 0: first code segment of a program, first word of a code statement, first record of a file. A few that start at 1: byte index of a SORT key, first record of an IMAGE dataset, first byte of a record in an FCOPY compare operation.

10. Unfinished Comment and "STRING TOO LONG"

If you forget to end a comment (with >>), SPL will ignore all source that follows until the next >> symbol. This can be very difficult to spot, since no error message occurs. One thing to look for is the P-counter column on your compile listing. After each statement, it should increase a little. If it doesn't, that line of code may be a comment. A similar type of error occurs when you forget to close a string with a terminating quote ("). SPL keeps scanning for a quote and eventually (several lines later) prints a STRING TOO LONG error message.

11. "TYPE INCOMPATIBILITY"

SPL does not allow type mixing in expressions. Type transfer functions are provided to make the data type of each operation unambiguous:

```
INTEGER LEN; LOGICAL FACTOR;
FACTOR:=LEN;  <<okay, no arithmetic, same size>>
FACTOR:=FACTOR * LEN;  <<ambiguous>>
                 ^
        TYPE INCOMPATIBILITY
FACTOR:=FACTOR * LOGICAL(LEN);  <<either this,...>>
FACTOR:=INTEGER(FACTOR) * LEN;  <<...or this>>
```

12. AND/OR versus LAND/LOR

If you want to err on the safe side, never use the AND/OR operators, only LAND/LOR (logical AND/logical OR). LAND and LOR (like XOR and NOT), are logical operators that combine 16-bit logical result that can be saved or tested:

```
LOGICAL L; INTEGER I;
LOGICAL PROCEDURE CHECK'BUF;...
L:=  I=5  LOR  I=50;
IF I=5 LAND CHECK'BUF THEN PROCESS'BUF;
```

Note that in the last example, when the IF statement is executed, the function CHECK'BUF is always called. AND/OR are not logical arithmetic operators; they are more akin to structure points like IF, ELSE, WHILE, UNTIL, etc.

```
LOGICAL L;INTEGER I;
LOGICAL PROCEDURE CHECK'BUF;...
L:=  I=5  OR  I=50;   <<invalid>>
               ^
       MISSING SEMICOLON
IF I=5 AND CHECK'BUF THEN PROCESS'BUF;
IF I=5 OR CHECK'BUF THEN PROCESS'BUF;
```

The use of OR in the assignment statement is invalid because OR does not produce an arithmetic result to be assigned. The IF examples show valid uses of AND/OR, but do not make clear the side effects.

Each time the first IF is executed, the function CHECK'BUF is only called if I=5; the proper interpretation of that statement is:

```
IF I=5 THEN
  IF CHECK'BUF THEN PROCESS'BUF;
```

Each time the second IF is executed, CHECK'BUF is only called if I=5; the proper interpretation of that statement is:

```
IF I=5 THEN PROCESS'BUF
ELSE IF CHECK'BUF THEN PROCESS'BUF'
```

13. BYTE Address and External Procedures

Although SPL gives you the ability to treat each 16-bit computer word as two independent 8-bit quantities (BYTE), the MPE routines used for input/output accept only word arrays. That is, the address in your stack for a READ or PRINT must always be an even byte boundary (left byte of a word). When you pass a BYTE array to one of these routines, SPL converts it into an INTEGER array and prints a WARNING:

```
BYTE ARRAY B(0:9);
PRINT(B(1),-5,0);    <<means PRINT(B(0),-5,0);>>
```

When the BYTE address specifies the leftmost (or even-numbered) byte of a word, the resulting INTEGER address refers to the same starting point in your stack. However, when the BYTE address refers to the right half of a word (odd-numbered), the starting address will always be rounded down to the next lower byte. The symptoms are that your messages print out with one character added at the beginning and one dropped at the end. To get around this problem, do your input/output into word arrays, and then move to byte arrays.

SPL always converts addresses by generating an "arithmetic shift" instruction that preserves the sign bit of the address. Unfortunately, this is not always correct. For positive addresses, a "logical shift" should be used, since any word address above 16K will generate a byte address above 32K (sign bit needed as a data bit). For negative addresses, use "arithmetic shift" to preserve the negative sign.

```
BYTE POINTER BP; ARRAY BUF(0:16);
@BP:=IF @BUF<0 THEN @BUF&ASL(1) ELSE @BUF&LSL(1);
```

14. NOWARN

The SPL compiler generates WARNING messages for a number of situations, some serious and some trivial (see #13). It is very tempting to use $CONTROL NOWARN to suppress these messages, but it can lead to costly errors. Some of the warning messages indicate potentially serious flaws in your program. The safest approach is to eliminate the cause of the warning (i.e., byte address

warnings can be eliminted by declaring integer arrays for input/output, and equivalencing byte arrays to them for internal manipulation).
Here are two cases where warnings can be very important:

```
<<Local PB array initialization>>

EQUATE MAX=3;
ARRAY TABLE(1:MAX)=PB:="SU","DE","FR","TO";
  WARNING  INITIALIZATION OUT OF RANGE ^
WHILE I<=MAX DO...  <<you forgot to change MAX>>

<<passing a value to a reference parameter, see #16>>

PRINT(10,2,0);   <<passes 10 as buffer address>>
        ^
  WARNING  EXPECTS REFERENCE PARAMETER
```

15. OPTION VARIABLE

One of the most difficult mistakes to detect is an error in calling an OPTION VARIABLE system intrinsic such as FOPEN. These routines allow you to omit parameters in the list (leaving only a comma to hold their place), and to omit completely unneeded trailing parameters. Unfortunately, it is very easy to miscount the number of commas between two parameters. If you make a mistake, the source program may still compile without a syntax error. The way around this source of errors is to maintain a sample call to each such intrinsic in an editor file. The sample should put each parameter on a separate line, so that there can be no confusion. In order to call the intrinsic, you add the sample text into your source program and modify it.

Here is a sample call to the FOPEN intrinsic:

```
filenum :=
  fopen (                 <<formaldesignator BA>>
  ,                       <<foptions         LV>>
  ,                       <<aoptions         LV>>
  ,                       <<recsize(-=bytes) IV>>
  ,                       <<device           BA>>
  ,                       <<formmsg          BA>>
  ,                       <<userlabels       IV>>
  ,                       <<blockfactor<256  IV>>
  ,                       <<numbuffers       IV>>
  ,                       <<filesize         DV>>
  ,                       <<numextents       IV>>
  ,                       <<initialloc       IV>>
  ,                       <<filecode         IV>>
  );
```

And here is an actual call created from this sample:

```
input'filenum :=
  fopen (input'name       <<formaldesignator BA>>
  ,1 <<old>>              <<foptions         LV>>
  ,                       <<aoptions         LV>>
  ,                       <<recsize(-=bytes) IV>>


,input'device         <<device           BA>>
,                     <<formmsg          BA>>
,                     <<userlabels       IV>>
,                     <<blockfactor<256  IV>>
,8                    <<numbuffers       IV>>
,                     <<filesize         DV>>
,                     <<numextents       IV>>
,                     <<initialloc       IV>>
,                     <<filecode         IV>>
);
```

Other OPTION VARIABLE intrinsics that should be called carefully are FGETINFO, SORTINITIAL, CREATE and WHO.

16. VALUE versus Reference

SPL allows parameters to be passed by VALUE or by Reference. In COBOL and FORTRAN, all parameters are passed by Reference, and this is the default parameter type in SPL. When a procedure has a reference parameter, it has access to the actual variable passed by the caller (not just a copy of the variable). That means the procedure can change the contents of the external variable, either intentionally (as when passing back results), or unintentionally (treating the parameter as a local variable). Obviously, only a variable name can be passed as a Reference parameter, not a constant or expression. VALUE parameters do not give the procedure access to any external data storage. A VALUE parameter is merely a local copy of the value passed as the parameter. Thus, VALUE parameters can be passed either variables or constants or expressions. The procedure can treat the parameter just as if it were an initialized local variable, but cannot pass back results in it (since it disappears when the procedure exits). Arrays, however, cannot be passed by VALUE.

17. Subroutines and FOR Loops

Both subroutines and FOR loops are dangerous constructs to use in SPL, because they load control data onto the top of the stack upon entry. If that data is not present upon exit, unpredictable side effects result, usually terminating in an ABORT for bounds violation or stack underflow.

Because subroutines provide a useful function that cannot be duplicated by any other SPL construct (i.e., they allow you to break the statements within a procedure into logical modules, while still referencing local variables), they should be used; but only if certain restrictions are followed. Never jump into or out of a subroutine with a GOTO. Never refer to the stack explicitly within a subroutine (i.e., no TOS, no SCAN, no MOVE with "sdec").

FOR loops can be easily duplicated with the WHILE statement and should, therefore, not be used. If you do use them, observe the same restrictions as for subroutines.

18. The Index Register

Many SPL programmers are tempted to use the index register as a variable. This usually leads to bugs that are very difficult to detect, because many innocent looking SPL constructs change the contents of the index register. Besides array indexing, they include: CASE statement, X <=Y <=Z, SWITCH statement, ASL (N), and allocation of local variables in procedures with muliple entry points.

19. Condition Codes

Many MPE intrinsics return status information only via the hardware condition code. If some other status result is provided, it should be used, since the condition code is a very fragile entity: almost every HP 3000 machine instruction changes it. Here is an example:

```
FILES(I):=FOPEN(FNAME,1);
IF <> THEN...   <<condition code changed by indexing of FILES
                  after FOPEN>>
<<The proper code for this problem is below>>
TEMP:=FOPEN(FNAME,1);
IF <> THEN....
FILES(I):=TEMP;
```

(Continued)

20. The Ultimate Error

When Compiling an SPL program, it is possible to destroy your source file if you miscount your commas. Suppose you intend to compile a master M with a textfile T:

```
:SPL T,,,M      (correct)
:SPL T,,M       (incorrect, uses M as listfile and erases
                 the contents.)
```

For more information on using SPL/3000 in commercial applications, please write to me at Robelle Consulting Ltd., #130-5421 10th Avenue, Delta, B.C. V4M 3T9 CANADA

"HYBRID" Information Retrieval Package

James Taylor
Parlimentary Subcommittee
Representative
House of Commons
London, England

The HYBRID system is an extension of ADHOC (Accessible Documentation for the House of Commons), a full-text retrieval system based on natural language searching. Both HYBRID and ADHOC do approximately the same job as the IBM STAIRS package-retrieving units of unstructured text through content-searching based on combinatorial logic.

HYBRID1 DOCUMENTATION: OPEN (DATA-BASE NAME) THIS COMMAND READS INTO THE COM—PUTER THE CONVENTIONS DECLARED FOR THE DATA-BASE IN THE ORIGINAL MAKE COMMAND. OPEN IS A NECESSARY PRE-CURSOR IN ANY SESSION OF ANY ORDER COMMAND EXCEPT MAKE (DATA-BASE NAME) OR CARRIAGE-RETURN AT COMMAND LEVEL, WHICH ENABLES YOU TO LEAVE HYBRID1 ETC AND RETURN TO THE MPE LEVEL.

HYBRID1 PERMITS YOU TO:
- OPEN (DATA-BASE NAME)
- MAKE (DATA-BASE NAME)
- SEARCH
- RETRIEVE
- VERIFY
- BROWSE
- FILE (OUTPUT-FILE NAME)
- COPY (OUTPUT-FILE NAME)
- INDEX
- UPDATE

HYBRID1 DOCUMENTATION: FILE (OUTPUT-FILE NAME) THIS COMMAND DOES FOUR THINGS:
- A) IT MEASURES THE VOLUME OF INPUT TEXT FROM AN EXISTING FILE.
- B) IT ACCEPTS NEW TEXT FROM THE TERMINAL (AN ALTERNATIVE TO A) ABOVE, AND MEASURES IT. (IN BOTH CASES A AND B) THE INFORMATION IS PROVISIONALLY FILED ON BUFFER)
- C) IT ASSIGNS FIXED-FIELD HEADERS TO ABSTRACTS, IF SO REQUESTED AT RUN-TIME.
- D) IT PLACES AN END-OF-FILE MARKER ("%") IF THIS IS MISSING.

HYBRID1 DOCUMENTATION: COPY (OUTPUT-FILE NAME) COPY CREATES THE RAW TEXT-FILE WHICH HAS PROVISIONALLY BEEN FILED ON BUFFER, AND SHOULD BE EXECUTED BEFORE RE-USE OF THE FILE COMMAND (TO AVOID OVERWRITING INFORMATION PREVIOUSLY FILED BUT NOT YET STORED ON A HYBRID TEXT-FILE.)

HYBRID1 DOCUMENTATION: INDEX (TEXT-FILE NAME = OUPUT-FILE NAME FROM COPY)
THE INDEX COMMAND CREATES A RAW-INDEX OF ALL WORDS NOT IN THE EMBEDDED STOP-LIST (ABOUT 120 COMMON WORDS - "A", "THE", "AND", "IS", ETC) WHICH WILL NEVER BE USED FOR RETRIEVAL AND WHICH WOULD THUS BE REDUNDANTLY STORED IN THE FINAL "INVERTED FILE" ON WHICH THE SEARCH COMMAND (SEE SEARCH DOCUMENTATION) OPERATES. INDEX WORDS BY SEE-SAW MERGING BETWEEN TWO OUTPUT FILES, AND DECLARES THE NAME OF THE FINAL RAW-INDEX FILE. THIS MUST BE NOTED FOR DECLARATION AT THE UPDATE STAGE WHICH FOLLOWS.

HYBRID1 DOCUMENTATION: UPDATE
UPDATE (IN HYBRID) DOES TWO MAIN OPERATIONS:
- A) IT MERGES THE RAW-INDEX WITH THE EXISTING REFERENCE-FILE (INVERTED FILE), CREATING A NEW ONE.
- B) IT APPENDS THE HYBRID1 TEXT-FILE (CREATED IN THE COPY COMMAND) TO THE MASTER TEXT-FILE. (NOTE - THIS WILL CHANGE IN HYBRID2 WHICH WILL DO WITHOUT A MASTER-TEXT FILE.)

IMPORTANT NOTE! THE HYBRID TEXT-FILE AND RAW INDEX ARE NO LONGER REQUIRED FOR SEARCHING/RETRIEVING AFTER THE UPDATE. HOWEVER, PLEASE DO NOT PURGE THESE FILES, NOR (IF APPLICABLE) THE ORIGINAL NON-HYBRID1 (E.G. EDITOR-CREATED ASCII FILES) WITHOUT ENSURING THAT THEY ARE STORED ON MAG-TAPE, BECAUSE THEY WILL BE NEEDED FOR HYBRID2, UNLESS OF NO ARCHIVAL VALUE WHATEVER (E.G. TEST-DATA).

HYBRID1 DOCUMENTATION: MAKE (DATA-BASE NAME)
NOTE: AT CURRENT DATE, ONLY IN INTERPRETER!
MAKE DECLARES THE CONVENTIONS AND NAME OF A NEW DATA-BASE, USING QUESTIONS AND ANSWERS.
- A) IT ASKS THE D-B CREATOR TO ASSIGN SPECIAL CHARACTERS (OR ACCEPT DEFAULT CHAR—ACTERS) FOR DELIMINTERS, FILE-ADDRESSING CHARACTER, ETC. OF THESE THE MOST IMPOR—TANT IS THE ABSTRACT DELIMITER ON WHICH SEARCHING IS BASED. THE WRITER TENDS TO USE A BLANK LINE; ANY ONE WHO DOES THE SAME SHOULD REMEMBER THAT IF THEY WANT EMBEDDED BLANK LINES WITHIN A SINGLE ABSTRACT, SHOULD PUT SPACES OR OTHER NON-PRINTING CHARACTERS.
- B) AT THE APPROPRIATE QUESTION, ALL INDEXING SHOULD BE DECLARED AS BY LINE NUMBER ("L").
- C) IF FIXED FORMAT FIELDS IN HEADERS ARE REQUIRED, THE USAGE MUST BE CHANGED IF

THE EARLIER BLANK LINE CONVENTIONS HAVE BEEN ACCEPTED (TO AVOID USING THE SAME DELIMITER FOR TWO DIFFERENT PURPOSES).

HYBRID1 DOCUMENTATION: SEARCH

A) SEARCHES ARE AUTOMATICALLY NUMBERED (FOR LATER RE-SEARCHING AND FOR REFERENCING HIT-FILES FOR RETRIEVAL) SEQUENTIALLY FROM 1, RESET TO 1 WHEN THE USER OPENS A NEW DATA-BASE.
A SEARCH COMMAND IS COMPOSED OF AN OPERAND (SEE BELOW)
OPTIONALLY FOLLOWED BY ANY NUMBER OF PAIRS OF OPERATOR OPERAND, SEPARATED BY SPACES.
THUS:
OPERAND OPERATOR OPERAND OPERATOR OPERAND (ETC) NINE LEVELS OF PARENTHESIS ARE ACCEPTED. (USERS SHOULD BE CAUTIOUS OF AMBIGUITIES IN USING BOOLEIAN OPERATORS.) (SEE BELOW IN MORE DETAIL ON OPERANDS AND OPERATORS)

HYBRID1 DOCUMENTATION: SEARCH (BOOLEIAN) OPERATORS

THERE ARE FOUR BOOLEIAN OPERATORS.

- -

OR	(ENLARGES A HIT-FILE)
AND	(RESTRICTS A HIT-FILE)
NOT	(RESTRICTS A HIT-FILE)
EOR	(MAY ENLARGE OR RESTRICT A HIT-FILE)

HYBRID1 DOCUMENTATION: SEARCH OPERANDS

OPTIONS ARE:

A) ANY WORD
B) ANY SET OF WORDS (PHRASE) SET IN QUOTATION MARKS OR OTHER PHRASE DELIMITER SET IN THE MAKE COMMAND.
C) ANY PREVIOUS SEARCH NUMBER PRECEDED BY A FILE-MARK: E.G. #2
D) ANY WORD-ROOT TERMINATED BY ? (THE WILD CHARACTER)
E) ANY SET OF (PRINTABLE) CHARACTERS CONTAINING EMBEDDED RELATIONAL OPERATORS:
THESE ARE <, <=, =, <>, > =, > E.G. D> 790205, AUX=TAYLOR, ETC.
F) AS WITH EO, BUT THE VALUE FOLLOWING THE RELATIONAL OPERATOR SET AT THE SPECIAL VALUES .MAX. OR .MIN.
E.G. D .MAX. WILL GIVE THE LATEST UPDATES OF AN APPROPRIATELY HEADED SET OF ABSTRACTS.

HYBRID1 DOCUMENTATION: RETRIEVE

IN RETRIEVE MODE THE MAIN (PROMPTED) CHOICES ARE:

A) THE HIT-FILE (E.G. #12 FOR THE TWELFTH SEARCH).
B) THE VALUE TO BE RETRIEVED: IN GENERAL "T" (FOR TEXT) FOR ONLINE RETRIEVAL, OR "R" (FOR REFERENCES) FOR THE LINE OR ABSTRACT-NUMBERS (DEPENDING ON THE MAKE SETTING) OF HITS WHERE THESE WILL TAKE TOO LONG TO POINT OUT, AND WHERE THE USER HAS THE FULL HARD-COPY.
"M" FOR MICROFORM WILL GIVE THE MICROFORM FRAME REFERENCES.

(NOTE: IN HYBRID1 THIS ONLY WORKS WITH THE KODAK "CAR" INTERFACE DEVICE FOR AUTOMATIC TRACKING.
TO THE REQUIRED FRAME-NUMBER. OTHERWISE IT IS NECESSARY SIMPLY TO READ THE "M=..." FIELD IN THE HEADER.

HYBRID1 DOCUMENTATION: BROWSE (FILE-NAME)

PERMITS INSPECTION OF TEXT FROM THE MAIN DATA-BASE (DEFAULT) OR FROM ANY TEXT-FILE, THE USER DECLARING THE RANGE OF LINE-NUMBERS, SEPARATED BY COMMA, TO READ TO END PUT LARGE NO.). (LINE NUMBERS MAY BE TAKEN FROM THE RETRIEVE OUTPUT.)

HYBRID1 DOCUMENTATION: VERIFY

PERMITS INSPECTION OF THE INVERTED FILE (MASTER INDEX) TO CHECK ON THE PRESENCE OR ABSENCE OF SPECIFIC WORDS, PHRASES, ETC.

HYBRID2 DOCUMENTATION: TELEMAIL

TELEMAIL IS AN ELECTRONIC MAIL FACILITY WHICH PERMITS THE USER TO DIRECT EITHER TERMINAL INPUT OR AN EXISTING FILE TO ANOTHER USER. TELEMAIL FEATURES INCLUDE:

UNREAD	(SUMMARIZES UNREAD INCOMING AND OUTGOING MAIL)
WHENREAD	(GIVES DATE/TIME WHEN INCOMING AND OUTGOING MAIL WAS READ)

WHOIS (STRING1 STRING2..) (GIVES TELEMAIL ADDRESSEE-CODES)
PRINT (MESSAGE OR FILENAME (S)) PRINTS AND FLAGS AS READ
SEND (FILENAME) TO (ADDRESSEE-CODES) (SENDS FILE(S), OR FROM TERMINAL)
ACCEPT (FILENAME (S)) REMOVES INCOMING MAIL FROM MAILBOX
HELP (LISTS OPTIONS OF TELEMAIL)

HYBRID2 DOCUMENTATION: EXTERNAL

EXTERNAL PERMITS INVOCATION OF CUSTOM ROUTINES (E.G. FOR CONVERSION OF A DATA-BASE INTRODUCED FROM ANOTHER SYSTEM, OR ADDITION OF ANY NEW FACILITY DESIRED BY AN INDIVIDUAL INSTALLATION)
SYSTAX IS EXTERNAL (PROGRAM NAME)

HYBRID2 DOCUMENTATION: DIRECT EXECUTION OF MPE COMMANDS BY: ANY MPE COMMAND CAN BE PRECEDED BY : FOR EXECUTION WITHOUT LEAVING HYBRID2.

HYBRID2 DOCUMENTATION: EXIT

THIS IS A SUBSTITUTE FOR C/R TO LEAVE HYBRID2, AND ALLOWS HYBRID2 TO BE CONTROLLED FROM FIXED-RECORD-LENGTH ASCII JOB-FILES (I.E. PROGRAMMATICALLY EDITABLE JOB-FILES).

HYBRID2 DOCUMENTATION: RESTART (PASSWORD)

PERMITS BY-PASSING OF PRELIMINARIES: LAST TEXT-BASE USED IS AUTOMATICALLY OPENED, SEARCH-NUMBER IS SET AT THE NEXT SEARCH AFTER THE LAST ONE FROM THE PRECEDING HYBRID2 EXECUTION.

Conversion From "HP" to "HP"

by: John A. Beckett
Southern Missionary College
Collegedale, Tennessee

As of July 1, the Southern Missionary College Computer Service department had two HP computers. One was a 2100-based HP 2000 with 32 ports and 47 MB of disc (RJE not implemented). The other was an HP 3000 Series II with 185 MB of disc, recently upgraded to 448 KB from 320 KB of memory. On July 5, as we completed a report on useage on both computers for Fiscal Year 1979, we started a feasability study to see if we should and could abandon the HP 2000 for an upgrade of the HP 3000. As of this writing (August 8), the HP 2000 is in the hands of another organization. This article is a description of what we went through during that time.

Before even identifying an actual buyer for the HP 2000, we determined that for an organization to have any reason to buy the machine, they would have to be currently running an HP 2000F. So we knew from the beginning there would be three major tasks involved in the project:

1. Converting our users on the HP 2000 (which included some administrative work for SMC, all 'Intro' students, and a few outside users we sell time to) to the HP 3000. Intro students were easiest. We just converted between semesters.
2. Converting our 800 BPI tapes to 1600 BPI. The 800 BPI drive on the HP 2000 was capable of being switched to the HP 3000 via a daisy-chain cable and switch, so this was possible.
3. Converting the buyer's HP 2000F applications to HP 2000.

We started with the tape conversion, as that was easiest to design and took the longest. The only interfacing required was to declare tape files as REC=1036,,U. The rest was simply waiting and changing reels, cleaning heads, etc. We discovered that at a room temperature of 75 degrees the 800 BPI drive would begin acting strangely after running several hours. Lowering the thermostat solved that problem.

Converting HP 2000F applications to our machine involved using a program formerly marketed by HP for use under MPE-C and MPE II. It didn't work under MPE III because of tape labels. DECOMP told us where the offending bit in AOPTIONS was in both FOPENS, and PATCH allowed us to fix it. We also discovered through our SE that tape labels can be switched off easily. (Ask your SE how to do it if you need to.) The program worked properly except where programs damaged by crashes on the HP 2000F were encountered. This problem required an extra trip to the user site to identify and purge the offending programs and then get a new HIB (similar to SYSDUMP Date=0). The buyers' people assumed responsibility for actual changes to programs. They gave us a check, packed up their "new" computer, and headed for Atlanta. (P. S. They did the move themselves, and it worked when they plugged it in. HP, take a bow for good gear!)

The big job was converting our HP 2000 users to the HP 3000. We found that IITOIII was not a very useful tool for this need because of two limitations: It was not good at retrieving things a user number at a time, and it truncated program statements to 72 characters. We had a version of FCONVERT we had previously converted from HP 2000F to HP 2000 format, and used it for almost everything instead. Its primary flaw was that it would abort if one attempted to convert a CSAVEd program. So we had to log in and un-CSAVE all programs on the computer.

The part of IITOIII we did use extensively was the documentation. This has an excellent list of differences between BASIC on the two machines. Some of these (ITM and PRINT ‡n USING . . .) have been added to BASIC/3000 since. Things that caused the most trouble:

1. SYSTEM statements. These give commands to the operating system, and are thus far different. In some cases we found equivalents (i.e., CALL WHOMS (A$) replaces SYSTEM A$, "TIME" where you are wanting a user name only). In others it was difficult (SYSTEM A$, "CAT-FILEX") or impossible (SYSTEMA, "BYE").
2. Accuracy. The double-precision algorithms we had used on the HP 2000 did not work on the HP 3000 because variables of type REAL aren't quite as accurate on the HP 3000. Much gnashing of teeth!
3. Performance. You can CHAIN on the HP 2000 faster than you can on the HP 3000, especially if you have been converted before a memory upgrade arrived. The obvious solution was to compile sets of programs that CHAIN among themselves.
4. Termtype problems. We had some terminals used on the HP 2000 for which no termtypes existed on the HP 3000. TDELAY2 was used to modify the amount of delays allowed for CR and LF to provide for these terminals.
5. Use of Common. This is documented by IITOIII.

In summary, we found the HP 3000 contributed library an excellent source of tools for the project — and would have found it quite impossible without them. I have vowed to review the library every few months from here on to see what other problems have come up for which somebody else has an answer, and to contribute the answers we have developed so others may utilize them.

HP3000 Series I Conversion

by: John C. Peterson
Comprecare, Inc.
Denver, Colorado

Many computer users are terrified by the though of upgrades and associated conversions. I would like to relate our recent experience in upgrading from our HP/3000 Series I to a Series III. COMPRECARE is a health maintenance organization in Denver, Colorado, with an enrollment of 55,000 members. Unlike a typical health insurer, an HMO provides a health delivery system for its membership which encourages frequent checkups, etc., and discourages waste of health care dollars on unnecessary procedures and hospitalization. Most of Comprecare's administrative operation flows through our data processing system which maintains enrollment, processes claims from physicians, pharmacies and hospitals and generates reports, which analyze the utilization of health care dollars.

The system is built on an IMAGE data base with 25 date sets. Total size of the data base is 80 megabytes with 16 months of history. All software is in COBOL and QUERY. Our Series I configuration consisted of one 7925 disk, a Memorex/BST 600 LPM printer and four Hazeltine terminals. We had this sytem for one year without any downtime and only a handful of system failures, all a result of the configuration of MPE tables and buffers (TBUF overflows, etc.).

Other than the relatively slow performance of the Series I, the only major system problem was the management of the date base with DBUNLOAD/DBLOAD. Just prior to installing the Series III, we attempted to increase the size of a large (250,000 entries) master date set with DBUNLOAD/DBLOAD. After 72 hours, we crashed with a tape error and had to restore the original date base. Then we used ADAGER/3000 developed by Alfredo Rego of Guatemala. The date set was successfully transformed in 12 hours.

Our upgrade to the Series III was a tremendous success; the upgrade also included the installation of a second 7925 disk and the installation of 4 additional Hazeltines. Prior to the installation, we performed three SYSDUMPS for redundancy of backup. The two HP CE's started the installation at 9 P.M. on Monday morning and finished at 2 P.M. Tests and diagnostics were run until Tuesday morning. Then we configured MPE III and did a RELOAD with the SPREAD option. By 11 A.M. Tuesday, we were live without any software or data conversion. We had done hardware failure later that week which the CE's quickly diagnosed as a bad ROM chip and repaired. As a result of the upgrade, our interactive data entry and updating has increased 3 fold despite the addition of 4 more terminal stations. Batch report generation time has been cut in half and in many cases the reports run 5 times faster. COBOL compile time has decreased as much as 50 times (from 49 minutes 20 seconds to 1 minute 3 seconds). The data base transformation with ADAGER, described above, ran in 5½ hours.

I realize that this procedure sounds very routine and that is exactly why I am so euphoric. There were no surprises or no conversions. HP just pulled out the Series I, plugged in the Series III, and we were up and running even in a mixed vendor environment.

HP3000 Job Scheduler

by: Clive Oldfield
The London Business School
London, England

Initially, I believe it would be helpful to give a brief outline on the salient points, that convinced us that we required some form of scheduling system for our H.P. 3000 computer system.

Currently our system comprises:

—HP 3000 Series III 1 mega byte.
—3 HP 7920 50 mega byte disc drives.
—1 HP 7925 120 mega byte disc drive.
—2 HP 7970E 1600 bpi tape drives.
—1 HP 7970B 800 bpi tape drive.
—1 HP 2617 line printer.
—1 HP 2613 line printer.
—32 port muliplexor (12 lines on dial-up).

on which we provide a computing service (Fortran, Basic, Ksam) to approximately 50 users both internally and externally.

After the initial installation, and once system experience had been gained it soon became apparent from the free-for-all that ensued, that the system lacked some desirable operational control functions in the following three areas:

— Enforcing job CPU resource limitations.
— Scheduling and providing notification of tape requests.
— Cancellation of jobs by users prior to execution.

The first item on the list was considered to be of primary importance as we found no useful mechanism which could be adopted to enforce certain job CPU resource limitations within different time slots. (e.g. Setting a maximum CPU time of 300 seconds during Prime Shift.) It is true that by using variations of different input priorities some form of control could be gained. Unfortunately this option is not totally secure as some measure of trust has to be placed upon the users to submit work with the correct parameters.

Due to our moderately active tape environment, we found that we needed some advance notification system, for our operations department, of tape requirements prior to access. It was also realized that if this facility was coupled with a scheduling system we could then overcome the tape and resource conflict that frequently occurred. In order to completely implement such a system, it was necessary to prohibit tape accesses from sessions.

One other drawback we found was that a user could not cancel a job once it had been streamed. In our experience, users ofter realize some alteration to the job file is necessary immediately after submission.

Since the introduction of the "SCHEDULER" system, further benefits have been seen and provided. The following list is an outline of the current facilities implemented:

— Imposing Maximum Job Prime Shift CPU Time.
— Controlled Tape Scheduling/Tape Setup Details.
— Job Cancellation by Users Prior to Job Execution.
— Controlled Time/Date Job Scheduling.
— Common User Interface for Submitting Jobs to MPE or Foreign Computer via MRJE.
— Line Editing Functions on Run Once Only Job Schedule File. (Replace: STREAM)

Another obvious advantage with a Scheduling system is in the event of a system failure, only the jobs previously executing are lost, if the system cannot be "WARM-STARTED".

The "SCHEDULER" package comprises two main programs (MONITOR: Program handling monitor and scheduling facilities. JASPER: User interface program for submitting jobs.) plus two accessory programs, all written in FORTRAN IV. The user is allowed to schedule batch work for running in accordance with the scheduling limits imposed by the installation's system management, on an immediate, specific time and date or every day basis. An example of scheduling limits that may be imposed is the trapping and deferring of jobs which request excess CPU time to that allòted to prime shift. Jobs which use tapes may also be trapped for operator control.

The facilities for tape handling are written in such a way that the operator can obtain details of what tapes a job will require before the job is scheduled for execution. Thus enabling more controlled allocation of resources and avoiding the starting of jobs which may later have to be suspended because a required resource is unavailable or

(Continued)

already in use by another process. This is provided by the inclusion of "COMMENT SETUP" statements, within a job file, which are copied to an operator inspection file when the job is submitted. Alternatively, these statements may be used to relay other pertinent information to the operator.

The monitoring routine, "MONITOR", only streams immediate run jobs instantly. The others are submitted either when the scheduled time arrives, or under the direction of the operator. Avoided is the situation in which a large "WAIT" queue is built up by the MPE "STREAM" facility, which is subject to total loss in the event of an unrecoverable system crash.

The programs comprising "SCHEDULER" are as follows:

1. MONITOR: Program for monitoring and submitting jobs.
2. JASPER: Program run by users for building and scheduling batch jobs.
3. BUILDER: Program for building necessary files.
4. JOBLIST: Program for inspecting the previous 24 hrs. job statements.

MONITOR

Facilities incorporated are:

1. Submission of jobs scheduled under JASPER.
2. Collection of routine statistics.

Optional

3. Interfacing with MRJE/3000 to handle scheduled work for other sites.
4. Processing of output from MRJE/3000 in non-standard HASP evironments when the normal facilities of MRJE will not function.

N.B. 3 & 4 are performed by SON processes to avoid delay to normal scheduling of H.P. 3000 work.

JASPER

Facilities incorporated are:

1. Scheduling of work for immediate or specified date and time.
2. Scheduling of work for every day runs.
3. Facilities for building and correcting errors in run-once-only job files, without leaving the scheduler. Files may be kept permanent for later submission.
4. Ability to subsequently modify date, time of run.
5. Ability to delete job from schedule file.
6. Provision for listing "pre-run" job information on job status, tapes etc. for owner or global.

Optional

7. Scheduling of work for MRJE/3000 submission.

SPECIAL REPORTS

Bug/Enhancement Poll

by Ross Scroggs

With this issue of the JOURNAL the Interface Committee of the HPGSUG initiates a polling procedure through which the members of the Users Group can indicate which defects in current HP software products critically affect the operation of their HP3000. Included in each issue of the JOURNAL will be a prestamped return card on which you will list your most critical bugs. The responses will be collected and forwarded to HP for their consideration. HP plans on using this information as one of the variables in the bug prioritization function. The results of each poll will be published in the subsequent issue of the JOURNAL.

Included on the poll card is space for an enhancement request. You should include a brief description of an enhancement you would like to see in a current HP software, hardware, service, or other product. These responses will be classified and distributed to the appropriate HP personnel. These requests, along with those received through other means, should provide HP with information concerning the direction they might want to take when planning future enhancements. These requests may also be used to formulate questionnaires that will address specific product areas.

HP will respond in subsequent issues of the JOURNAL as to how they are using the information gathered in the polls and will provide general indications as to what direction they are taking with regards to specific problems or products.

The Interface Committee urges you to complete the enclosed poll card as soon as possible so that the results can be provide to HP in a timely manner.

Please note that this poll does not replace the normal bug reporting mechanism, you should promptly report all new bugs through the regular bug reporting procedure. The purpose of this poll is to determine the number of people affected by a particular bug.

Editor's Note: The BUG/Enhancement Poll Card is found on the back cover of the JOURNAL. Clip and mail today! (No Postage is required.)

New Software Reporting Service

by: Illene Birkwood
HP General Service Division
San Jose, California

July 1 marked the first day of General Systems Division's new problem reporting service - the Software Tracking and Reporting System (STARS). This system is the result of months of effort aimed at providing you with more responsive service. This new IMAGE/3000 based system will enhance our ability to monitor software bugs, enhancement requests and documentation errors.

Features

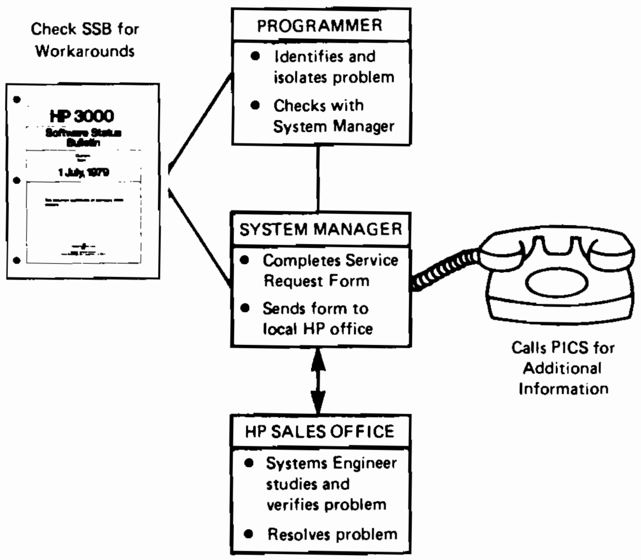
- Problems resolved faster by moving the source of resolution closer to you - your local HP office is now the focal point for solutions.

  As each bug is received at the local HP sales office, it is reviewed by a Systems Engineer familiar with your application. If the Systems Engineer finds an immediate solution - perhaps the "bug" was the result of someone misunderstanding the documentation, or the bug has already been reported and a workaround exists - he contacts you right away and resolves the problem.
- A streamlined reporting mechanism which ensures that problems which impact your operation receive the correct level of priority within HP.
- Workarounds and known problems communicated more effectively through a new, easy to use format of the Software Status Bulletin.

In response to the input received at last year's Users Group meeting in Denver, we are publishing the Software Status Bulletin in complete form once each calendar quarter. Between quarterly issues, a bi-monthly (twice a month) update will be published. The update will be cumulative and will contain all new problems and any existing problems on which the status has changed since the quarterly issue.

To help you locate your problem in the Software Status Bulletin, a keyword index has been added. Problem reports are assigned a keyword which is used to group them with problem reports of a similar type. Thus, by looking up "Bounds" in the keyword index, it is possible to locate all problem reports relative to bounds violations.

A brief one-line summary of the problem has been added to the keyword index. (For example, COBOL HANDLING OF A "READ INTO RECORD AREA" WHEN AN END OF FILE IS ENCOUNTERED). By scanning new problem reports in the latest update to the Software Status Bulletin, you will be able to quickly locate any problems similar to your own.


Check SSB for Workarounds
HP 3000 Software Status Bulletin
1 July 1979
PROGRAMMER
• Identifies and isolates problem
• Checks with System Manager
SYSTEM MANAGER
• Completes Service Request Form
• Sends form to local HP office
Calls PICS for Additional Information
HP SALES OFFICE
• Systems Engineer studies and verifies problem
• Resolves problem

SOFTWARE PROBLEM REPORTING SERVICE

Dealing with Software Problems

To keep an operation functioning smoothly, it is important to know how to handle problem situations. When a contingency plan is in place for every situation, production continues to flow smoothly around the problem. Since software problems are a fact of life in every computer system, let's talk about ways to circumvent them and keep your operation functioning.

There are several steps which need to be taken whenever a software bug is suspected:

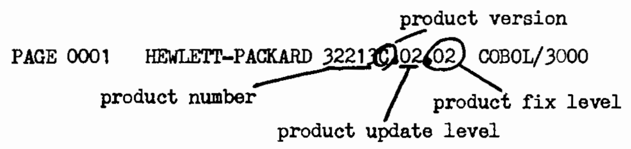
1. Identification
2. Isolation
3. Documentation
4. Verification
5. Classification
6. Resolution

If you encounter any difficulties as you use the procedures described in the following text, your System Manager may call the Phone-In Consulting Service (PICS) for help and advice.* The System Manager provides the interface between HP and your staff - in this way, no confusion or duplication of effort ocurs at your site.

**PICS is only available to customers with Customer Support Service (CSS). Customers with a Software Subscription Service may receive assistance on a time and materials basis.*

1. Identification

When a problem occurs, you should identify which HP software product is involved by providing the product number, version, update level and fix level. This is necessary because the problem may only be duplicated with a particular level of the subsystem and operating system. For example, if the problem occurs with COBOL, the identification information can be found from the header on a compilation listing.



If there is doubt about the version and level, consult the most recent issue of the COMMUNICATOR or Software Status Bulletin that describes the software update code release being used, or call the Phone-In Consulting Service.

The origin of software problems is complicated by the interaction of the operating system and the various subsystems. A call to the Customer Engineer will provide your System Manager with direction on what action will be taken when the operating system is the suspect. The Customer Engineer is also the source for determining which hardware problems might also manifest themselves as operating system problems.

2. Isolation

The next step is to isolate the problem. (However, if you cannot isolate the problem, the Phone-In Consulting Service (PICS) is available to discuss what on-site services are required.)

(Continued)

(New Software Service Continued)

A. Change the environment and execute only selected software modules.

B. Determine if the software module has been executing in the past. If it has, determine what changes have been made to the module since the last successful execution.

C. Segment the software module into smaller programs and execute each independently to isolate the problem.

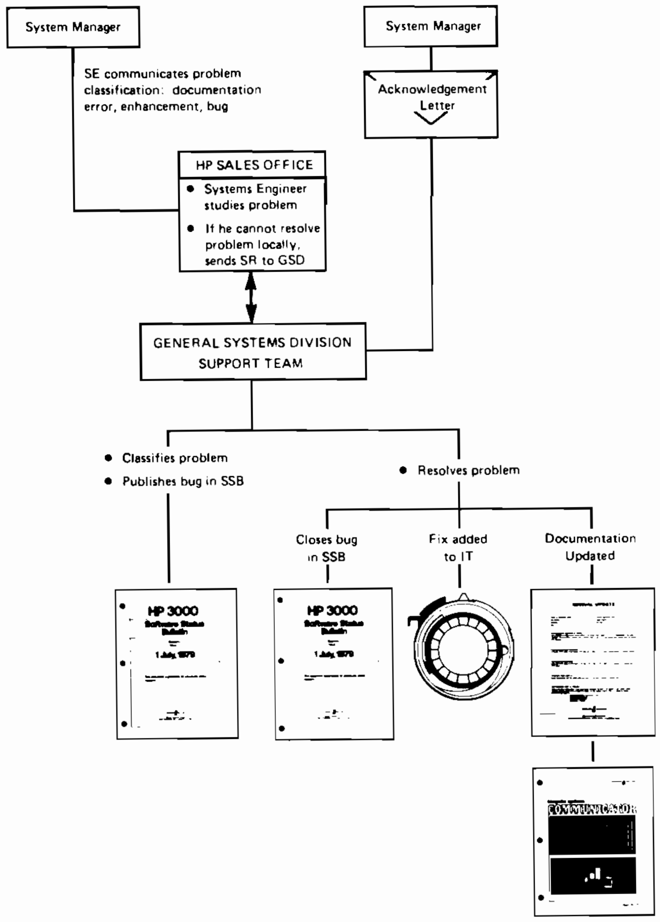
3. Documentation

When submitting your Service Request, include any documentation which will enable the HP engineers to reproduce the problem. Efforts to resolve problems are greatly enhanced when the problem is well documented. Your Systems Engineer can tell you about the type of documentation that is required.

4. Verification and Workaround

Having identified the bug and verified that it can be reproduced, check the Software Status Bulletin to see if the problem has been reported and a workaround is available. If you find the problem reported in the Software Status Bulletin, General Systems Division is aware of the problem and steps are being taken to solve the problem.

If you cannot find the problem or workaround in the Software Status Bulletin, then call your local Phone-In Consulting Service center and find out if more recent information is available.


System Manager
SE communicates problem classification: documentation error, enhancement, bug
System Manager
Acknowledgement Letter
HP SALES OFFICE
Systems Engineer studies problem
If he cannot resolve problem locally, sends SR to GSD
GENERAL SYSTEMS DIVISION SUPPORT TEAM
Classifies problem
Publishes bug in SSB
Resolves problem
Closes bug in SSB
Fix added to IT
Documentation Updated
HP 3000
HP 3000

"BEHIND THE SCENE"
SOFTWARE PROBLEM REPORTING SERVICE

If the problem is unique, complete a SERVICE REQUEST form and send it, together with any supportive documentation, to the local sales office to the attention of the SR Monitor. (Copies of the Service Request form can be printed on your system with the FORMGEN program in the PUB group of your SYS account, or you may use the two copies available in each Software Status Bulletin.) Your Systems Engineer will review the form and verify the problem. He may call you at this point to obtain additional information, or to let you know that the problem duplicates a similar problem and is already being worked upon. He may be able to resolve the problem at this stage - if not, he will work with you to help find a workaround until a permanent solution is achieved. It is very important that a suitable workaround be found, since permanent fixes are not released until they can be incorporated in a regularly scheduled release of the installation tape.

5. Classification

Problems forwarded to General Systems Division by your Systems Engineer for resolution are recorded by the Service Request Monitor at the Division. Each Service Request is entered in the IMAGE date base and a unique identifier is assigned. This identifying number is included in the acknowledgement letter which is sent to the System Manager. If you have any questions about your Service Request, always use this number when checking the Software Status Bulletin or talking to your Systems Engineer.

After your Service Request has been classified by HP engineers at General Systems Division, the status of your request is communicated to your Systems Engineer. He then contacts you by letter or phone, letting you know the classification of your problem. The Service Request is also published in the Software Status Bulletin.

Not all the Service Requests submitted to General Systems Division are listed in the Software Status Bulletin. Only software design errors and documentation errors are listed. The remaining Service Requests are duplicates of previously reported problems, misunderstandings of the way in which the system operates, problems that cannot be duplicated or hardware problems. Your Systems Engineer will communicate with you, letting you know how your Service Request has been classified.

6. Resolution

When the software problem is resolved, the information will appear in the Software Status Bulletin, and the corrected software will be distributed on a regularly scheduled release of the Installation Tape.

NOTE: A complete description of the new STARS system is included in "Guide to a Successful Installation" which can be obtained from HP Sales Offices, part number 30000-90135.

TIPS & TECHNIQUES

AN APOLOGY!

Our zest in pursuit of the publication schedule met head on in July NEWSLETTER with our equal zest for accuracy. The editing of Tom Harbron's article, "Helpful Hints for System or Account UDC's", was so "successful" no one, especially Tom, recognized it! We apologize for the mistakes of the past and have learned a valuable lesson for the future. Tom's article is reprinted below. Again, we extend our apologies!

Helpful Hints for System or Account UDC's

by: Tom Harbron
Anderson College

The 1906 release of MPE3 permits User Defined Commands on a System or Account wide basis. Issue #20 of the Communicator gives the basic information, but not the whole story. The following discussion pertains to System-wide UDC's, but similar problems may occur with Account-wide UDC's.

The first problem occurs when a user logs on in an account other than SYS. Immediately following the welcome message, he receives an error message saying there was a security violation while attempting to access the System-side UDC file. The problem is that the Command Interpreter opens the UDC with Locking. However, the standard security on the SYS account doesn't allow that. The simplest solution is to RELEASE security on the UDC file.

The second problem occurs when an attempt is made to remove the file from the catalog. The SETCATALOG command without parameters seems to do nothing. The correct command is SETCATALOG;SYSTEM. This removes it from the catalog; however, all users who logged on prior to the command still have the file open. Thus attempts to purge or alter it are futile until all such users have logged off. The only solution is to wait (usually until the next day).

UDC File/Peanuts; SYS

by: John S. Borden, Jr.
Baltimore Gas & Electric Co.

After reading Tom Harbron's "Helpful Hints for System Account UDCs" in Issue 3 of the NEWSLETTER, in which he pointed out some problems, I think the following items will help those with the same problems.

1. If you wish to place the system wide UDC in the SYS account it is true that default security prevents its use by "Normal" users. RELEASING the file solves this problem, but pretty much leaves it vulnerable to anything. This can be remedied by placing a lockword on the file name in the setcatalog command. e.g.

   :SETCATALOG UDCFILE/PEANUTS;SYSTEM

2. The second problem Tom brings up is changing the contents of the system wide UDC file. Our approach is as follows:

   Assume the system wide UDC file is named SYSUDC1. If we wish to modify it we make a copy called SYSUDC2, modify it as desired and then make it the new system wide UDC. Any users signing on from then on get the new one while users signed on at the time the mods are made keep the old one for the remainder of the session.

Remote Hardwired Terminals

by: John Scott
Conestoga College of Applied Arts and Technology
Kitchener, Ontario

We have video terminal remote-hard-wired to the control centre 1500 ft. away. The terminals are presently running at 2400 band, using a cable with 11 sets of wires, each set individually shielded; so far, we have found no problems caused by loss of data during transmission.

Bell 212A Modem Support On The HP3000

by: Tom Black
Hewlett Packard
Cupertino, CA

The Bell 212A or equivalent is an extremely good modem for use with the ATC or ADCC. It provides 1200 baud full-duplex operation using a single telephone line and gives improved performance over the Bell 202S modem.

The use of 202S is no longer recommended. 212A's cost only a little more, give better performance and overcome all of the installation problems we have encountered using half-duplex operation. With the ATC, there is no need to use Option 002 (ATC option for half-duplex mode); Option 001 is used (ATC option for full-duplex mode) saving $1200 per ATC.

We encourage your use of 212A's instead of 202S's, the net result will be vastly improved satisfaction.

Calculating Optimum Disc Space File Size

by: Larry Stinson
VALTEK, INC.
Springville, UT

How many times have you designed a file and wanted the optimum disc space file size? This can be calculated as per the steps given which is the same as the Block program in the Contributed Library.

- Calculate number, including fractional part, of records in one sector.
  A=256 / (bytes/record) or
  128 / (bytes/record)
- Calculate Blocking factor as follows:
  BF=A*⌈(1/ (A - ⌊A)
  where ⌈ ≡integer greater than or equal to following value and ⌊ ≡integer less than or equal to following value.

Shrinking Stack Size In COBOL Programs

by: Lloyd Julick

This subroutine for COBOL programs was developed by Lloyd Julick (current address unknown), to shrink stack size after calling a dynamic subprogram that uses a large amount of stack space.

```
00000 0         $CONTROL SUBPROGRAM,
                SEGMENT=SHRINK
00000 0         BEGIN
00000 1         PROCEDURE SHRINK;
00000 1         BEGIN
00000 2         INTEGER S,Z,ACTSIZE;
00000 2         EQUATE MINSTACK=1000;
00000 2         INTRINSIC ZSIZE;
00000 2         PUSH(S,Z);
00002 2         Z:=TOS;
00003 2         S:=
00004 2
00004 2         IF Z> (S+MINSTACK)
00007 2            THEN
00011 2            ACTSIZE:=ZSIZE(S+MINSTACK);
00016 2         END;
00000 1         END.

PRIMARY DB STORAGE =%000
NO. ERRORS=0000;
PROCESSOR TIME =0:00:00;

SECONDARY DB STORAGE =%00000
NO. WARNINGS=0000
ELASPED TIME=0:00:14
```

LIBRARY CORNER

Infobase Changes

by: Wayne E. Holt
Library Committee

The Library Committee is pleased to announce that Release 06 of the Contributed Library will be ready for distribution in mid-October. Tape reproduction will take 8-10 weeks; distribution will be coordinated by the Executive Director, Rella Hines. The new version will include many new contributions, including software from the last SCRUG meeting, and the International Meeting in Lyon, France.

The most notable change in the latest release concerns INFOBASE account. The Committee has been working for several months to evaluate the effectiveness of this account. There has been quite a bit of feedback since the introduction of this concept in June 1978 with release 04, both pro and con. The following are the changes as approved by the Excutive Board at the Boise meeting:

1. **Reduce the redundancy of the INFOBASE account.** Under the original plan for an information account, a separate group for each type of file structure (IMAGE, KSAM, or Sequential) was established. Each group was self-contained, including supporting software.

   Aside from the maintenance problems, it is difficult to justify from a "use" perspective. The Users are not taking advantage of the features of the more advanced file structures as was originally thought. Therefore, only a single, sequential version of the information file will be kept. It, and its supporting utility program, will be located in PUB.-INFOBASE.

2. **Continue Multiple Accounts.** The "Multiple account" approach to the Library seems to be working. Several large contributions have been received that would never have fitted the old single LIB concept. An extension of this will be implemented this summer. The LIB 2000 software will be established as an account on the HP3000 Library. All soft-ware in the LIB that was earlier contributed from the HP2000 will be moved to the new account. The LIB2000 account will not be distributed unless specifically ordered by the User from the Executive office.

3. **Improvement of the GINFO Files.** The current general information files in INFOBASE are simply inadequate to meet the needs of either the first-time or experienced user. A genuinely comprehensive guide will be prepared to meet these needs.

   The new guide will be automatically printed with a copy of the abstracts to ensure that the User is aware of the structure and uses of the Library. Additionally, the guide will be prepared in such a manner that it will be appropriate for printing and distribution to potential members and other interested parties. If not complete by Release 06, this guide will be a priority item for Release 07.

4. **Documentation files.** The format of the documentation files has long been an undecided issue. The Committee has received many suggestions and has decided upon a format that closely parallels the one developed by SCRUG. All DOC files in the LIB account will have this format on Release 06.

5. **Store SOURCE and JOB (or) PROG, not both.** Wherever possible the Library has traditionally offered the User the Source code, program code, and a job stream to compile and test. This constitutes an "overkill" and wastes valuable space on the tape. The committee has decided to keep only one or the other. SOURCE and JOB are the preferred method. If both are submitted, the Central Library Site will purge the PROG version in PUB. This will save about 25 000 sectors of storage at this time.

6. **Text Compression.** By replacing blank lines and multiple spaces with special codes, the ASCII files in SOURCE, DOC, and JOB can be reduced to about half their original size. The Central Library site will write programs to perform the compress/uncompress functions as dictated by the Users. If not completed by Release 06, it will be a priority item for Release 07.

   The following data represents a benchmark performed at the Central Library Site:

		Uncompressed	Compressed
Disc:	@.DOC.LIB	13594	6236 (in sectors)
	@.SOURCE.LIB	75517	29430
	Total	89111	35666
Tape:	@.DOC.LIB	1750 ft @ 1600 bpi	720 ft @ 1600 bpi
	+@.SOURCE LIB	2860 ft @ 800 bpi	1180 ft @ 800 bpi

The tape lengths are estimated using the program TAPELEN, which calculates based upon 1 K blocks.

It is expected that 4 K blocking will significantly alter the results. By way of comparison, the entire uncompressed Contributed Library consists of 169,707 sectors and at 1 K blocking consumes 3330 ft @ 1600 bpi and 5450 ft @ 800 bpi.

These changes represent the Committees' effort to deliver a better, more usable Library to the User Group. However, their efforts are wasted unless you provide the necessary feedback and criticism that is fundamental to constructive change. Let the Committee know what you think regarding this latest revision. Note also that the structure of the Committee is changing. Effective September 1, I am stepped down as Chairman of the Committee and Editor of this corner of the Journal. Lilla Solberg, the current Chairman Pro-Tem, is taking my place. She will bring fresh insights and energy to the position, and I hope to see a great deal of progress in the next two years.

PRINTER REQUIREMENT SURVEY

FEATURE

Rank each of the following features in order of importance: (1 being most important, 2 being second most important, 3 being third most important, etc.)

	RANK
Ease of Operation	
Forms Handling (including dual tractors, paper jam detection, paper stacking capability)	
Low Noise	
Price	
Print Quality	
Print Versatility (including multiple print sizes, languages and graphics)	
High Reliability	
Low Cost of Ownership	
Compact Size	
Other (please specify)	
Other (please specify)	

SPECIAL FORMS USAGE

How many types of forms do you use (including standard computer paper)?

____

____

Do you use special forms in the following widths (check all that apply)?

____ Less than 8½"
____ 8½" to 15"
____ 15" to 16"
____ 16" to 17"
____ Greater than 17"

Do you use forms in the following lengths (check all that apply)?

____ Less than 4"
____ 5" to 6"
____ 7" to 9"
____ 9" to 11"
____ 11" to 12"
____ 12" to 18"
____ Greater than 18"

What percent of your output is on special forms?

____ Less than 5%
____ 5% to 10%
____ 10% to 20%
____ 20% to 40%
____ 40% to 60%
____ Greater than 60%

APPLICATIONS INFORMATION

Indicate by check (✓) your present and future application areas and interest.

APPLICATION AREA	PRESENT	FUTURE	APPLICATION AREA	PRESENT	FUTURE
Accounting Payroll Checks Invoices Reports/Statements Other (specify)			Marketing Order Processing Document Quality Reports Business Graphics Other (specify)		
Manufacturing Inventory Reports Stock Flow Documents Receiving/Shipping Documents Other (specify)			General Program Listings Rough Draft Reports		

TYPE OF:

System(s) in use currently?

Name ____________________

Quantity ____________________

Printer(s) in use currently?

Name ____________________

Quantity ____________________

Number of printers per system in data center? ______

at remote location(s)? ______

Number of pages of output per day? ______

Peak day? ______

What is your satisfaction level with your printer(s)?

____

____

____

Fold Here

Return Address

____

____

____

____

Hewlett-Packard
Boise Division
P.O. Box 15
Boise, Idaho 83707
ATTN: Steve Richardson

Clip along dotted line

BUG/ENHANCEMENT POLL

NAME: ______________________ **COMPANY:** ______________________

PHONE: (   ) ______________________ **ADDRESS:** ______________________

TELEX: ______________________

Please indicate the BUGS that critically affect the operation of your HP3000. Include any number of bugs but indicate only those with greatest impact. The bugs should be identified by their Known Problem Report (Service Request) number as found in the most recent Software Status Bulletin. Please include BUGS with an "open" status only.

Please include one brief ENHANCEMENT Request (hardware/software/service/other) you would like HP to address. Be brief and include a Keyword that classifies your request as specifically as possible, e.g., 7925 Disc/COBOL/CE support.

KEYWORD: ______________________

REQUEST: ______________________

HP General Systems Users Group
Empire Towers
7300 Ritchie Highway
Glen Burnie, Maryland
21061, USA

RELLA M. HINES, EXECUTIVE DIRECTOR
(301) 768-4187

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 83
GLEN BURNIE, MD.

LINFORD HACKMAN
VYDEC, INC.
9 VREELAND RD.
FLORHAM PARK, NEW JERSEY
07932, USA

Clip along dotted line

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS **PERMIT NO. 8** **GLEN BURNIE, MD.**
POSTAGE WILL BE PAID BY ADDRESSEE

HP General Systems Users Group
Empire Towers
7300 Ritchie Hwy.
Glen Burnie, MD. 21061