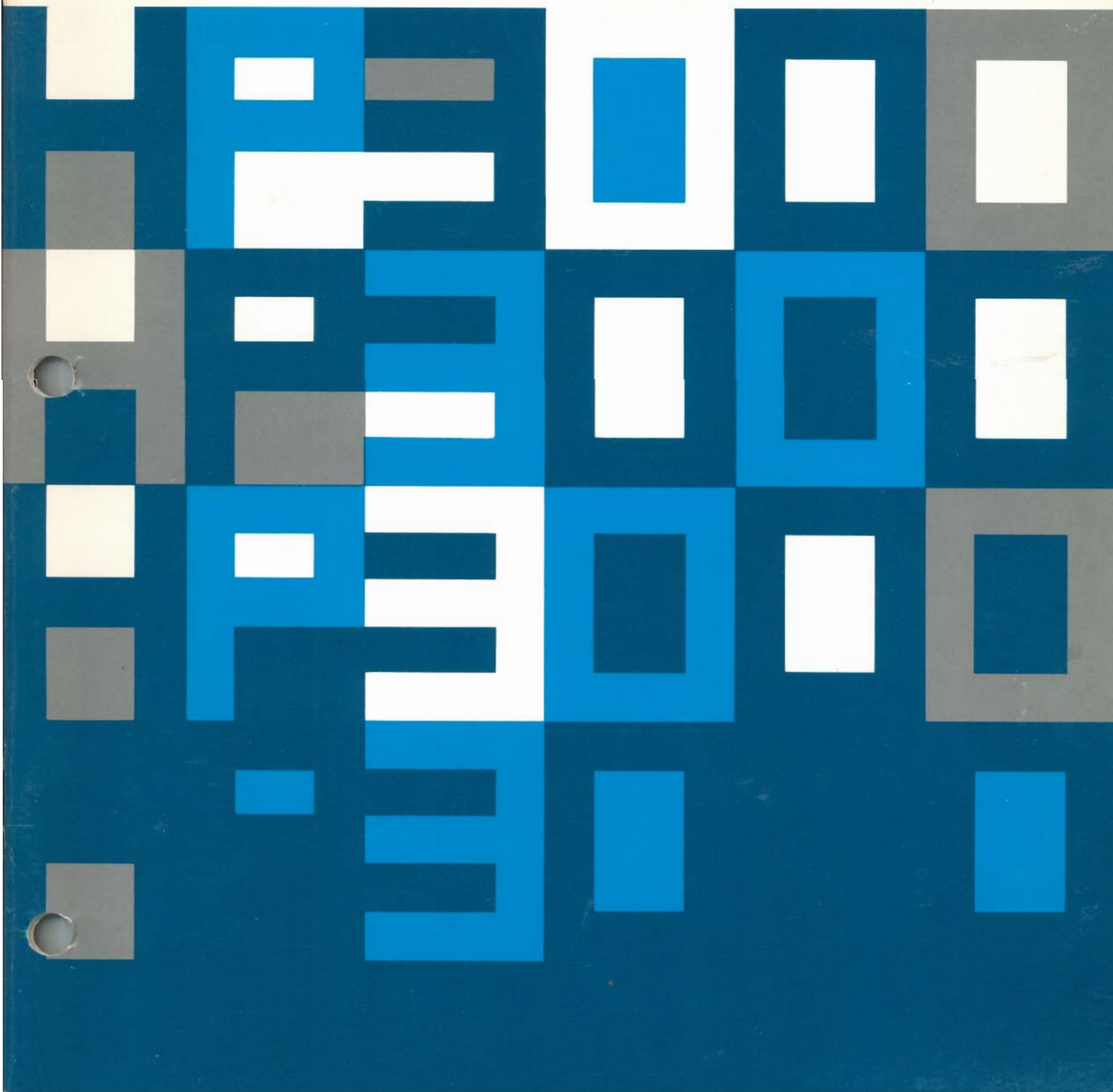
ARTICLES
MFG 1932
CIS 1928

HEWLETT PACKARD

COMMUNICATOR

ISSUE NUMBER 22



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Table of Contents

Editor's Note

We won't go as far as handing out cigars, but we are very proud of the latest addition to the family of HP 3000 computer systems. Named the Series 30, this new product is completely compatible with its elder and larger siblings, the Series III and the Series 30, yet it has the capability for standalone applications. You'll find an article discussing the Series 30 on page 4.

The 19 articles in this issue cover a wide variety of topics. We hope that all of the articles will interest you, but we do want to point out three articles in particular. "The Secret of Up-to-Date Documentation" (page 11) tells you exactly how to get the documentation service that's best for you. "STARS-- New Software Reporting System" (page 15) specifies a new procedure for bug reporting. And your participation is invited in "IMAGE Database Schema Survey" (page 44).

In addition to articles, this issue of COMMUNICATOR 3000 contains the College Information Systems (CIS) note files, date code 1928, and the MFG note files, date code 1932.

The next issue of COMMUNICATOR 3000, which will be published soon, will contain the latest Catalog of Customer Publications.

Editor
COMMUNICATOR 3000
HP General Systems Division
19447 Pruneridge Avenue
Cupertino, CA 95014

Introducing the New HP 3000 Series 30

"The Series 30 is fully software compatible with the larger members of the HP 3000 family."

by Chosen Cheng, General Systems Division

Hewlett-Packard Company has announced a new computer system as the third member of its HP 3000 family of business computers. Named the Series 30, this new system offers the same features and capability as its predecessors, the Series 33 and Series III.

Priced at $49,750, the Series 30 complements the mid-range Series 33 and top of the line Series III. Availability of the Series 30 is 14 weeks.

The Series 30 is an entry-level HP 3000 system for standalone business data processing applications. But its compact size, ease of installation, HP 3000 Distributed Systems Network software and remote console capabilities make it especially suited as an economical HP 3000 station in a network of HP 3000 systems.

The HP 3000 Series 30 is packaged within a small cabinet measuring only 24" by 36" by 18". The cabinet is accompanied by a separately packaged system/maintenance console and system disc. The system runs on single phase 110V-120V, 60Hz power (200V-240V, 50Hz power internationally). The small size and low power requirements of the system are made possible by use of HP's proprietary Silicon-on-Sapphire (SOS) technology.

While the Series 30 must be installed according to Hewlett-Packard guidelines for site preparation, the system will typically require no special air conditioning. The standard system configuration with console and 20 Mb disc generates only one 4650 BTU/hour, about the same as a medium sized copier. The Series 30 does not require the classic raised-floor, air-conditioned data center, and has been designed to operate over a broad range of customer environments.

The Series 30 is fully software compatible with the larger members of the HP 3000 family, the Series 33 and Series III. All HP 3000s use the MPE III operating system, and applications written on any of these HP 3000 systems can be run on any other without reprogramming, recompiling or relinking.

The HP 3000 Series 30 Computer System.
Shown here are a system processor unit, a system/maintenance console and a system disc drive.

The Series 30 incorporates a system self-test feature and remote system verification program (R.S.V.P.) to enable Hewlett-Packard service personnel to provide a high level of service and support at the customer site and from remote locations. The capability to allow control of the system from a remote system console by simply initiating the RSVP facility over a dial-up phone line is particularly important in a distributed data processing environment with many remote stations.

Hewlett-Packard's RJE/3000 communications product is also available on the Series 30 and Series 33, extending HP DSN capabilities to IBM mainframes as well.

All communications are handled by the new Intelligent Network Processor (INP), Hewlett-Packard's front-end communications processor, which also employs HP's SOS technology. The INP is a computer in its own right, providing the capability of off-loading communications control from the CPU. This enables high performance levels of local transaction processing during concurrent data communications to other systems.

A minimum system configuration for the Series 30 includes 256 Kb error-correcting semiconductor memory, CPU, 1 Mb flexible disc, four asynchronous terminal ports, system/maintenance console, 20 Mb system disc, and eight I/O expansion slots. A maximum configuration would contain 1024Kb memory, 960Mb high-speed disc storage, up to 32 terminal ports, with 4 magnetic tape drives, and 2 line printers. Up to 2 communication lines can be added, with each line replacing 4 terminal ports.

With the introduction of the Series 30, the HP 3000 family of compatible business systems for distributed data processing sets a new standard for the 1980's.

To Upgrade or Not to Upgrade

It's more than just a question of additional capacity.

by Dave Butt, General Systems Division

When considering whether to upgrade your current system or not, performance is often used as the measure in determining the necessity of upgrading. Questions such as "Would a larger system help me get my jobs completed faster?", or "Have my applications outgrown my capacity?", while addressing important issues, do not take into consideration all of the benefits of upgrading.

One of the other possible reasons for upgrading could be to take advantage of the latest technology. Using the HP 3000 Series I as an example, when upgraded to a Series III the main memory changes from error detecting core memory to error correcting semiconductor memory. In this way, technological advances have made the system more reliable.

A second example of technological advancement is in the area of communications. The ability to have system to system communications is only available on the more recent HP 3000 computer systems. Thus, the owner of an early member of the HP 3000 family might want to upgrade to a Series III to take advantage of this capability.

Expanding capability is a another reason for upgrading. As technology develops, peripherals as well as new systems will be produced using the most current technology. These peripherals will offer new capabilities for the user. But to take advantage of these capabilities, some peripherals maybe have to be designed in ways that allow them only to be offered only on the newer systems (due to MPE enhancements, interface standards, or other considerations). Peripherals such as the 2621A/P low cost terminals and the 2619A 1000 line per minute printer are two examples of peripherals which are not offered on earlier systems.

At the same time that hardware advances allow the user access to additional capabilities, software advances give the user additional flexibility and better people productivity. Improved diagnostics, more system management facilities, logging, and other aids that make the system easier to use are all examples of software advances which have been developed on the newer systems.

Thus it can be seen that the decision to upgrade is not simply a question of additional capacity. Higher reliability, expanded capabilities, access to new peripherals, and improved ease of use are all reasons for upgrading to another system. The user must take all of these factors into account before the true value of upgrading can be understood.

Add-On Memory Price Reductions

The price per megabyte of memory on the HP 3000 continues to drop.

by Rich Edwards, General Systems Division

Random Access Memory (RAM) add-on memory prices for the HP 3000 Series III, Series 33 and Series 30 systems have been been reduced as of August 1, 1979.

Product/Description	Was	8/1/79	% Reduced
30078A (128 kb for Series 30/33)	$4,000	$2,500	37.5%
30008B (256 kb for Series III)	$8,000	$3,750	53.1%

This represents $15,000 per megabyte for the Series III, and $20,000 per megabyte for the Series 33.

The price of a megabyte of memory on the HP 3000 has been lowered significantly over the past five years from $320,000 for core-based memory in 1975 to today's $15,000 for 16K RAM semiconductor memory.

The configuring of larger memory systems is one cost-effective way of growing your HP 3000 system as your needs grow since the addition of incremental memory can result in shortened response time and throughput improvements. Contact your Hewlett-Packard sales representative to discuss specific application and memory needs.

The Costs of Adding Incremental Memory

Only additional memory boards are needed to expand both the Series III and Series 33 systems up to one megabyte; no additional controller boards or power supplies are necessary.

For field expansion beyond one megabyte in the Series III, you should order the 30418A Memory Expansion Kit which contains two memory boards (512kb total) plus a memory controller board. The price of this upgrade kit has been reduced to reflect the new $15,000 per megabyte memory price:

Product/Description	Was	8/1/79
30418A (1536kb Memory Expansion Kit)	$17,000	$9,000

For customers ordering additional memory beyond the standard 256kb in a Series III (32435A), the system options have been adjusted:

Product/Description	Was	8/1/79
32435A option 505 expands memory to 512kb	$8,000	$3,750
32435A option 506 expands memory to 768kb	$16,000	$7,500
32435A option 507 expands memory to 1024kb	$24,000	$11,250
32435A option 509 expands memory to 1536kb	$41,500	$20,250
32435A option 511 expands memory to 2048kb	$57,500	$27,750

Thus, a two megabyte Series III now costs $132,750, down %18 from $162,500 prior to August 1st.

The Secret of Up-To-Date Documentation

Three different subscription plans are explained.

by Mary Eicher, General Systems Division

An accurate, up-to-date documentation library is a valuable component of any efficient computer system. The responsibility for keeping your HP 3000 documentation current is one that you, the HP 3000 user, share with Hewlett-Packard. The better we understand our respective roles, the more successful we will be in assuring that your documentation is as accurate and up-to-date as possible.

HP's Role

It's HP's task to generate documentation and, once a manual exists, to see to it that you know about the manual. We do this in two ways. First, if the manual is related to a new product and you buy that product, we automatically send you a copy of the manual. Second, if the manual (like USING FILES, for example) doesn't deal specifically with a new product, we announce the new manual in the HP 3000 Communicator.

It is also HP's responsibility to keep the information in the manual accurate. This is why from time to time we update a manual. Here again, we use the HP 3000 Communicator to notify you when a manual has been updated. Seeing to it that you receive the updated information is a task that falls under your area of responsibility.

Your Role

When it comes to maintaining your documentation library, you have three options. You can:

1) Subscribe to either of HP's support services: Customer Support Service (CSS) or Software Subscription Service (SSS). You automatically receive a complete set of manuals and one copy of every update and new edition issued for as long as you retain the service.

2) Subscribe to HP's Software Notification Service (SNS). You receive the Software Status Bulletin (twice monthly) and the HP 3000 Communicator (quarterly). Manuals and updates are listed in the Communicator so that you can check for new documentation and order the manuals you need.

3) Subscribe to HP's Manual Update Service (MUS). For a monthly fee based on the manuals you choose to maintain, you automatically receive updates and applicable new manuals as they are issued.

With the increasing amount of HP 3000 documenation, the above options allow you to get the documentation service that's best for you. The following table and paragraphs summarize the benefits provided in these plans.

	CSS/SSS	MUS	SNS
Manual Updates	X	X	
Manual New Editions	X	X	
Software Status Bulletin	X		X
HP 3000 Communicator	X		X
Additional Support Services	X		

Support Services

Subscription to one of the HP support services entitles you to a wealth of services besides assistance with documentation maintenance. These services include: Phone-in Consultation Service (PICS), Systems Engineering assistance, and software and firmware updates. Your HP Sales Representative can help you select the support service product best suited to your requirements.

Notification Service is included as part of both CSS and SSS. Note also that MUS can be used to maintain additional sets of manuals at your facility, as CSS and SSS each provide maintenance for only one set of documenation.

Software Notification Service

The Software Notification Service provides you with two periodical publications: the Software Status Bulletin and the HP 3000 Communicator. Issued twice monthly, the Software Status Bulletin lists software bugs and explains temporary work-around solutions.

An HP 3000 Communicator issue is published with each periodic release of the Installation Tape. (There may also be issues of the Communicator published between IT Tape releases.) The Communicator contains articles of interest, "unofficial" note files in the case of a corresponding IT Tape, and a catalog of available documentation. This catalog identifies all user manuals relating to the HP 3000 products and the date each was printed and last updated.

You can use the documentation catalog in the Communicator to identify manuals which have changed. Then you can order these manuals by using the special order form provided in the Communicator. (Note - It is not possible to order only updated information; the entire manual must be ordered.)

Manual Update Service

The Manual Update Service lets you select the documentation, in manual sets, which you wish to maintain. Update information and new editions relative to the manuals you select are automatically mailed to you as they become available.

MUS is recommended for additional sets of documentation at your facility not covered by either CSS or SSS. The following table shows MUS manual sets and the monthly fees for each. MUS must be ordered for a minimum of one year.

HP 3000 Manual Update Service

Manual Sets	Monthly Fee
HP 3000 Manuals:	$40.00
Overview (4 manuals)	
MPE (11 manuals)	
Utilities (4 manuals)	
Data Management (7 manuals)	
Programmer Manuals:	16.00
MPE Pocket Guide	
EDIT/3000 manual	
Using Files	
MPE Intrinsics	
Compiler Library	
SORT-MERGE/3000 manual	
Segmenter manual	
Debug manual	
MPE Commands	
FCOPY/3000	
Error Messages	
Data Management Manuals:	
DBMS/3000 (2 manuals)	4.00
KSAM/3000	2.00
VIEW/3000 (4 manuals)	5.00
Language Manuals:	
COBOL/3000 (2 manuals)	4.00
SPL/3000 (3 manuals)	7.00
RPG/3000 (2 manuals)	3.00
BASIC/3000 (4 manuals)	5.00
FORTRAN/3000 (2 manuals)	3.00
APL\3000 (2 manuals)	5.00
Data Communications:	
DS/3000 (2 manuals)	4.00
RJE/3000	2.00
MRJE/3000	2.00
MTS/3000	2.00
Manufacturing:	
EDC/3000 (2 manuals)	4.00
IOS/3000 (2 manuals)	4.00
MRP/3000	2.00
SPC/3000	2.00
Miscellaneous:	
Scientific Library	2.00
SIS/3000 (4 manuals)	7.00
CIS/3000 (2 manuals)	4.00

STARS – New Software Problem Reporting System

New, streamlined procedures for reporting software problems.

by Babs Brownyard, General Systems Division

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 ablility 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 solution.
- 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 (SSB).

Software Status Bulletin

The Software Status Bulletin (SSB) is published in complete form once each calendar quarter. Between quarterly issues, updates will be published twice a month. 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 a problem in the SSB, a keyword index with a brief one-line summary of the problem has been added. By identifying the symptoms of your problem you can scan the SSB and locate problems similar to yours.

Problem Resolution

The following is a summary of steps that System Managers should take in resolving a software problem.

1. Assist the user to isolate and identify the problem.

2. Check the SSB to see if the problem has been reported and if there is a workaround.

3. If you have trouble isolating the problem, or cannot find the problem or workaround in the SSB, the next step will depend upon the type of support to which you subscribe.

    a. If you have purchased Customer Support Service, call your local Phone-In Consulting Service (PICS) for the latest information, advice and debug help. The PICS SE will help you determine whether on-site assistance is needed. Installations with Additional Site coverage should work through the central site where the System Manager should reproduce the bug and then proceed with the problem resolution.

    b. If you have Software Subscription Service (SSS), you should decide if on-site assistance is needed. You may purchase on-site consultation on a time and materials basis.

4. If the problem still appears to be an unreported bug in HP software, fill out a Service Request (SR) form.

5. Gather all supportive documentation.

6. Send the SR and documentation to the SR Monitor at your local HP sales office.

Remember, the speed with which a problem can be resolved is directly dependent on how well the problem is isolated and how reproducible it is, the accuracy of the SR, and the quality of the accompanying documentation.

STARS Procedure

Upon receipt of your SR, the local field support team reviews and verifies the problem. They attempt to reproduce the problem, check locally and with GSD for existing information on the problem, and then try to resolve it or arrive at a workaround. If the problem is the result of user misunderstanding or a known problem with a workaround, the SE will contact you with the solution.

If the problem is a new software problem, a documentation error or enhancement request, the SE completes a verification form and forwards it to General Systems Division for resolution. The SE will then work with you to find a workaround. This is an important step, as permanent fixes are not released until they can be incorporated into an Installation Tape.

When your SR is received by the GSD support organization, it is immediately assigned a unique reference number. The number is included in a short acknowledgement letter sent to you with a copy to your SE. All correspondence and inquiries regarding the SR must reference this number.

Your SE will be notified as to the classification (new problem, duplicate report, hardware, etc.) and disposition of the problem, and will contact you within five days of hearing from GSD.

When the problem is fixed by HP General Systems Division, and after this fix has successfully passed the normal testing and Quality Assurance procedures, then this fix will be distributed on an Installation Tape. This information will also be reflected in the SSB.

Service Request Form

The accompanying figure (p.21) is an example of a completed SR. The following is a brief explanation of this form.

1. S.O. Ref. # - This space is used by the local HP office verifying the problem to assign an identifier for tracking and filing purposes.

2. SR # - This space is for the number assigned to the SR by the SR Monitor.

3. Customer Report # - This space is provided for your System Manager to assign an identifier of his own to the SR for tracking and filing purposes.

4. Submitted by - name of submitter (user).

5. Firm name - name of company.

6. Division - name of company division.

7. Date - date of problem occurrence.

8. Company address.

9. Specific location of system in customer's facility.

10. System Model - 3000 Series I, II, or III, 30, 33.

11. Operating System - version of operating system on which problem occurred.

12. Product Name - name and version of specific product in which the problem occurred - e.g., COBOL C/02/00.

13. Problem Description - Put a detailed description of the problem in this space. Use the other side of the form if you need more space.

14. Supportive Documentation - list of additional documentation describing or duplicating problem. See next section for recommended documentation tools.

Three sources of Service Request forms are available to you.

1. SR forms may be printed on your system with the FORMGEN program in the PUB group of your SYS account.

2. Each issue of the Software Status Bulletin contains two copies of an SR form which you can duplicate for additional copies.

3. Tablets of pre-printed forms are available from your account Systems Engineer upon request.

Supportive Documentation

Additional documentation needed to duplicate the problem should accompany the Service Request (SR). Complete and accurate documentation is critical for GSD to respond quickly to the SR. Incomplete documentation will generally result in a "cannot duplicate" response from the SE.

Suggested and required supportive documentation are as follows:

- Any short programs or routines that demonstrate the failure.

- A compiled listing of the program(s) with a compilation and preparation map (PMAP). (REQUIRED when user program is involved.)

- A listing of the actual execution showing the indicated problem (REQUIRED).

- A STREAM file that will reproduce the problem, or a menu of commands and input/output for reproducing the problem. (REQUIRED when user program is involved.)

- A tape or diskette with the STREAM file, program source, USL, RL, SL, or program files and any data files needed to reproduce the problem. The originator's name and mailing address must be put on the tape to insure its return. (REQUIRED when programs larger than 50 lines of code are needed to isolate the problem.)

- If the problem was a system failure, and no isolation is possible, then a complete memory dump (cold dump) is required.

Documentation Tools

Documentation tools for submitting Service Requests are provided by two files in the PUB group of the SYS account. The first is a STREAM job file to help you give the information Hewlett-Packard needs to help with MPE problems. The second tool is a program file which will reproduce Service Request forms for reporting problems, documentation errors, and enhancement requests.

1. DUMPJOB.PUB.SYS

   The first file (STREAM job) called "DUMPJOB" will:

   a) Generate a Service Request form.

   b) Obtain a copy of the loadmap from the file LOADMAP in the PUB group of the SYS account. This is an absolute MUST item for the Operating System Specialist to attempt to analyze a cold dump listing.

   c) Obtain a copy of your system I/O configuration by doing a SYSDUMP to a null device ($NULL). This particular program will then ABORT after obtaining the I/O listing. Note that this section of the program will abort as part of its normal operation.

   d) You must also run DPAN2.PUB.SYS to obtain a listing of the cold dump of the problem on the HP3000 Series III (DPAN for pre-Series II).

   CAUTION: If the cold dump to be taken using this STREAM file was not physically taken on the host machine, the LOADMAP and I/O configuration generated will be incorrect. Proper administrative measures must be taken to assure that the LOADMAP and I/O configuration are the correct ones for the cold dump in question.

This STREAMed job logs onto the account SYS using the MANAGER user. Passwords must be removed from this account/user or the STREAM file DUMPJOB must be modified using the EDITOR subsystem.

2. Generating SR's from FORMGEN.PUB.SYS

The second file (program file) is called FORMGEN. This program generates twenty copies of the Service Request form for reporting software problems, documentation errors, and enhancement requests. The program output is automatically directed to a line printer in the device class LP. The output can be directed to another list device by using the formal file designator LIST. Example:

```
:FILE LIST;DEV=$STDLIST
```

The above example would direct output to the session/job output device, which should be a hardcopy terminal.

Enhancement Requests

Enhancement requests are submitted in the same manner as problem reports (on an SR form). HP continually enhances its products and considers all the suggestions submitted by its customers. While every effort is made to comply with suitable suggestions, HP is under no obligation to implement a request. Enhancement requests do not appear in the Software Status Bulletin.

HEWLETT PACKARD

S.O. Ref. # 54 (1)

HP service request form

SR # _____ (2) *(HP only)*

Customer report no. 1 (3) Date 3-10-79 (7)

Submitted by John Doe (4)

Firm name XYZ Company (5) Division Food Products Div. (6)

Street 100 Main St. (8) System Mgr. John Manager

City, State, Zip Newport Beach, CA 92660 Phone no. (714) 640-3000 ext. 25

Specific location of system in plant Bldg. 50 (9)

System model 3000 Series III (10) Operating System B.00.00 (11)

(e.g., 3000/III) *(e.g., AMIGO A.01.01)*

System Serial No. 123456 Product Name COBOL C/02/00 (12)

(e.g., BASIC A.00.05) v uu ff

Problem description *(include environment, symptoms, what you were trying to do, what went wrong and any other information that might be helpful).*

(13) Run a COBOL program which does an "ADD 1" to a field defined as PIC S99V9 comp value 12. The result of the add was -11 instead of +13.

Supportive documentation included with report.

MEDIA	DESCRIPTION
1600 bpi magtape	COBOL program (14)

Rev date: 2/20/79 *5955-1727*

Example of a completed HP Service Request Form.

Removal of the SHUTQ and OPENQ Spooler Parameters

by Steve Zink, General Systems Division

In order to make a simpler and friendlier operator interface, all of the spooling commands were rewritten for the 1918 Installation Tape. Among the changes was the removal of the SHUTQ and OPENQ parameters. There were several reasons for this change:

1. When all spool space has been exhausted, an automatic SHUTQ occurs on all spaced devices and device classes.

2. When the :STOPSPOOL command is used, a SHUTQ is performed on the ldev or deviceclass named.

3. Because device classes can now be spooled independently of devices by ldev, queues can be opened whether a specific ldev is spooled or not. For example,

```
:STARTSPOOL LP
:STOPSPOOL 6
```

   now results in an OPENQ for all DEV-LP references.

4. SPOOK can be used to purge spoolfiles while it outputs them to tape to be retrieved later when the system is as active. The following sequence will answer all user needs:

```
:RUN SPOOK.PUB.SYS
>FILE T;DEV=TAPE
>OUTPUT @.@;*T;PURGE
>EXIT
:STOPSPOOL 6
:JOBFENCE 14
:OUTFENCE 14
```

   The user can now drain the system of remaining jobs and then coldstart or warmstart the system. After bringing the system up, the user can run SPOOK to recover his output files:

```
:RUN SPOOK.PUB.SYS
>FILE T;DEV=TAPE
>INPUT @.@;*T
>EXIT
```

   SPOOK can be used while the system is running, thereby eliminating the need to drain the system.

On the a future Installation Tape, the operator commands will be extended with the ability to alter active spoolfiles by ldev number or by devicefileids.

A Word on Delimiters in DEVICE Byte Strings

by Steve Zink, General Systems Division

Currently, any non alpha-numeric character except the slash (/), period (.), asterisk (*) or pound sign (#) can be used as a valid terminator for a DEVICE string in relation to the FOPEN intrinsic. However, in order to extend the capabilities of the file system, and so as to impact as few existing programs as possible, the semi-colon (;) as a delimier will be redefined in the future to mean that more parameters will follow. ANY EXISTING PROGRAMS USING ";" AS A DELIMITER IN THE DEVICE BYTE STRING IN FOPEN SHOULD BE CHANGED TO SOME OTHER VALID DELIMITER. It is suggested that blanks, carriage return or null 0's be used.

Restoring from Floppies

by Dean Kelley, Neely Cupertino

Series 30 and Series 33 customers who use flexible discs (floppies) should be aware of an important difference in the SYSDUMP and RESTORE procedures.

When SYSDUMP writes to floppies, the first two floppies contain MPE and related files. SYSDUMP always starts with a new floppy to write user files, usually the third floppy.

A problem can occur on a restore when the customer is restoring from floppies containing a SYSDUMP. If the customer starts with the first floppy, then the message "not a store format" will be generated because MPE is not in a STORE/RESTORE format. The customer should either go directly to the third floppy, or, if not sure how many floppies were used for MPE, start with the first, then second, etc., until the error messagae is not received.

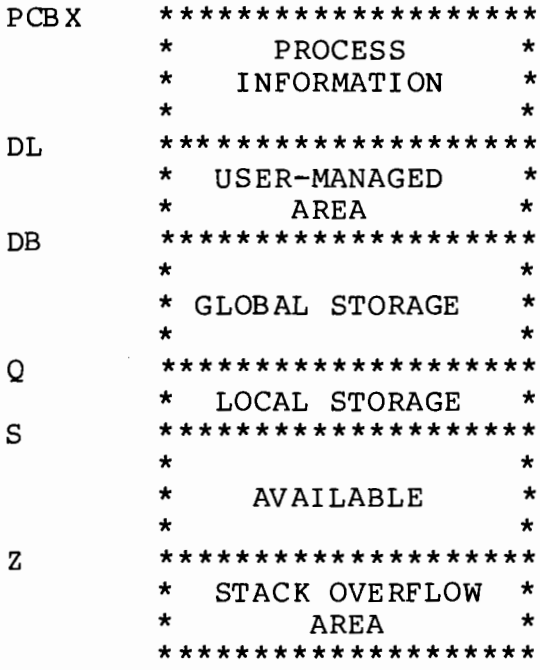
Use of STACK= and MAXDATA=

"Let the system expand the stack as necessary."

by Mark Cousins, Neely Cupertino

One of the major causes of poor system performance is misuse of the STACK= and MAXDATA= parameters on the :PREP, :PREPRUN, and :RUN commands. We will describe the stack in detail and then present methods you can use to prevent trouble.

When a process is initiated, the stack is set up as follows:



The PCBX (Process Control Block Extension) contains information about the process which needs to be present in memory when the process is running. This information includes directory pointers, the DSTs for any allocated extra data segments, and file control blocks (FCBs). This area of the stack is inaccessible by the normal user, and is initially about 1200 words long. As more files are opened, the system may attempt to expand this area. The area from DL to DB is a user-managed global storage area and is not accessed by MPE. The compilers use the 20 words from DB-1 to DB-20, and VIEW stores its screen buffers and control information here also. BASIC uses the DL area for file buffer allocation and parameter passing.

The area above Z is reserved for the system so that a stack overflow can be properly processed if it occurs in critical system code, such as FOPEN. The extra area will be granted, the FOPEN will be allowed to complete, and at that point the process will be aborted. This is to guarantee that a stack overflow will not occur while a process is SETCRITICAL, which would otherwise cause a system failure.

Rules and limitations:

A. Since a stack is really just a data segment, and since the maximun size of a data segment on the 3000 is 32,764 words (caused by byte addressing limitations), the area from the start of the PCBX to the end of the stack overflow area cannot exceed 32,764 words. This implies that the user's area will be somewhat smaller; in fact, the maximum size of the area from DL to Z is 31,232 words.

B. The user can limit the maximum size of DL to Z with the MAXDATA= parameter on the :PREP, :PREPRUN, and :RUN commands. Note that this is not the initial size of the stack, but the limit to which it can grow. At least MAXDATA words of storage are set adide in virtual memory, and the user is guaranteed to be able to expand the stack to this limit. Exceeding it causes a stack overflow interrupt and abort.

C. The size of the area from Q (initial) to Z is controlled by the language compiler, the STANDARD STACK SIZE parameter in SYSDUMP, or the STACK= parameter in :PREP, :PREPRUN, and :RUN commands. Generally, the size calculated by the language compiler is sufficient, and then STACK= parameter can be left off the command. If you specify STACK=, STACK words of storage are reserved immediately (IN ADDITION to DL to Q (initial)) at run-time).

D. The area available to the system for the PCBX is calculated by subtracting MAXDATA from 32,764, and then subtracting 128 (the size of the overflow area) from the result. Since MPE promises you that you can get MAXDATA words of storage, it must use this number as its limiting factor in calculating PCBX size. Recall that the file system uses the PCBX for file control blocks and that, if many files are to be opened concurrently, the area may need to be expanded to accommodate more FCBs. If the expansion cannot occur because MAXDATA+new PCBX size+128 would exceed 32,764, file system error 71 (too many files open) is returned to the calling program and the FOPEN fails. You can force the control blocks out of your stack and into an extra data segment with the ;NOCB parameter on the :RUN command, but this will force the file system (and hence IMAGE and KSAM) to operate more slowly. A better solution would be to reduce MAXDATA, if this is possible. This is usually the case since many customers set MAXDATA to its limit value as a matter of course.

E. Many users believe that setting MAXDATA arbitrarily high does not "cost" anything. As shown above, it can cause problems with the file system. Additionally, the virtual memory area for a stack is determined from the MAXDATA parameter. Changing the virtual memory size can only be done during a RELOAD, and while running out of virtual memory may not cause a system failure, it could cause the application to abort and therefore be just as frustrating as if the system had crashed.

F. Setting STACK too high can really cause performance problems, in that the process may be consuming far more memory for its stack that it is actually using.

In summary, do not use the STACK= parameter; instead, let the system expand the stack as necessary. You will need to include the MAXDATA= parameter whenever your program (or an intrinsic called by your program) will be dynamically adusting the DL or Z registers. This includes all programs that use VIEW, as well as any BASIC-compiled programs that use IMAGE or MPE files. Unfortunately, there is no sure-fire way to estimate what MAXDATA should be, but through experimentation and experience you will find an appropriate value for each program. VIEW generally requires about 3,000-6,000 words more than the program's data area. Most application programs will run with a MAXDATA of about 10,000 words, although SORT performance can be improved by raising MAXDATA to about 25,000 words.

Minimizing Head Contention

Some basic guidelines on disc space utilization.

by Mark Cousins, Neely Santa Clara

Good disc management includes optimization of free space areas and elimination or reduction of head contention. Head contention can be defined as excessive seeking caused by alternately accessing different areas (files) on one disc. In this situation, each physical read or write will cause the disc heads to transverse the disc surface, and since this can take up to 45 milliseconds (worst case, vs. virtually instantaneous transfer if no head movement is involved), performance begins to fall off dramatically.

Head contention can be caused by user requests for transfers from or to disjoint areas on the disc, or by MPE interleaving its disc requests with user requests. It is usually not possible to combine files such that no contention will occur, so the next best solution is to place files that will be accessed "together" on separate devices. This also holds true for MPE's disc requirements.

Therefore, the user should attempt to place directory files on devices separate from the data files that they relate to. Additionally, a dedicated system disc will yield great benefits in that memory management requests will be satisfied more quickly and such requests will not interfere with user file access. Archival files can be stored on the inner cylinders of the system disc with no loss in performance, and this will result in better space utilization on the user discs.

These functions can be controled with the :STORE and :RESTORE commands, or with the contributed program MOVE. Additionally, once the file placement is defined and set, system reloads should be accomplished via RELOAD, option ACCOUNTS, followed by :RESTORE @.@.@. RELOAD will recover all free space into one contiguous block, and :RESTORE can be forced to place files back on the same devices on which they were created.

Device class management is also important. A system disc is of little use if it is also a spool device. As a general guideline, place directory files and active program files on the fastest discs, while spoolfiles, source, and sequential files should be allocated to the slower devices. This can be guaranteed by establishing device classes for the various types of devices and then using these classes in the build statements for the various files.

It can be argued that, in a multiuser transaction processing environment, disc management of the sort that has been described is to little avail, because of the randomness and unpredictability of disc requests. While this may be true for certain periods of activity, not managing the disc guarantees that you have the problems of head contention.

Tips on :QUANTUM

Setting the timeshare quantum.

by Mark Cousins, Neely Cupertino

The :QUANTUM command is one of the commands that the System Supervisor can use to help "tune" the HP 3000 on-line. Proper use of the command can add greatly to system performance; on the other hand, misuse of the command can cause the system to behave erratically.

The :QUANTUM command serves two functions: one, to set the maximum time that a process in a circular queue will have control of the central processor before the Dispatcher re-examines the ready list, and two, to set the minimum priority levels for the circular queues. In this article, we will discuss how to set the timeshare quantum.

Normally, processes on the HP 3000 compete with one another for use of the CPU. This "competition" is controlled by a system function known as the Dispatcher, whose mission in life is to run the highest priority process that is ready to run. This process is constantly changing - and the change is also controlled by the Dispatcher. This is done to ensure that no one process monopolizes the CPU. However, the System Supervisor can control how long any one process can use the CPU before the Dispatcher switches to some other process; this time is known as the "quantum".

How does one determine what the correct quantum is? That depends on the system load, the amount of memory installed, and the nature of the processes that are executing. There are some rough guidelines:

- The less memory available, the longer the quantum should be. The reason for this is that switching processes will likely involve some swapping on small memory systems, so the less often the Dispatcher switches processes, the less swapping there will be (to a certain extent).

- "Characterize" the overall use of the system. Is it a transaction processor, or a number cruncher, or some of both (i.e., development)? The quantum for each type will be different. For the dedicated transaction processing system, with a high degree of I/O activity, most processes will use the CPU for a longer period to perform a calculation. In this case, the quantum should be set relatively long (600 - 800); most of the time, processes will consume only a small fraction of this before

they voluntarily give up the CPU to read from the terminal, but the time is available for the few instances when they need more CPU. Number crunchers, on the other hand, tend to use their full quantum each time they get control of the processor. Quanta on these systems should be set somewhat shorter (300 - 500) so that short transaction processes are not unnecessarily penalized by highly calculation-bound processes. Development systems tend to be a mix of both types and should have the quantum set somewhere in the middle, about 400 - 600.

A note on quantum setting: don't expect to see a change in performance if you change the quantum by ten milliseconds. It turns out that the quantum used by the Dispatcher is always an even multiple of 100, calculated as:

True quantum: =INTEGER (input quantum/100 + 2) * 100

So that if you set the quantum to 400, the Dispatcher will actually use 600; also, there will be no difference between a quantum of 400 and one of 499. The values recommended above take this into account; use these values in the :QUANTUM command.

The kind of effect the quantum can have was demonstrated to me not long ago. I was on-site analyzing performance and noticed the ready list (from :SHOWQ) was quite long - about seven processes. I then noticed that the quantum was set at 250, which is very low for a transaction processing mix. I raised it to 700 and the ready list cleared immediately; the processes were no longer waiting for the CPU and response time improved. Another time I was copying a tape and noticed it seemed to be going very slowly. By setting the quantum to 100 my copy process got dispatched much more often and because I never consumed my full quantum, the copy speeded up instantly. Of course, this was probably at the cost of the other users on the system.

Don't be afraid to change the quantum dynamically based on what is running in the system. It may be more advantageous to set the quantum longer during the day for the online users, and shorter at night when batch is running. It can really make a difference.

CIS/3000 Software Update

CIS/3000 HP32902A.00.02
Date Code 1928, N00N902A.HP32902.SUPPORT

A. ENHANCEMENTS.

1. CISREG has been enhanced to print negative image codes on abnormal conditions.

2. Working-Storage in CISBAR has been enhanced to include Copylib members SCH-MSTR, STU-DET, CRS-MSTR, and CRS-SEC.

B. CORRECTIVE SOFTWARE CHANGES.

All of the following known problems have been corrected in CIS/3000.

1. SR #5440 - The SCHOOL-ID field was truncated to two characters when format number 93 was processed by CISBAT, this is corrected.

2. SR #4537 - In CISREG the incorrect error message "STUDENT NOT ON FILE" was changed to "STUDENT NOT IN SCHOOL."

3. SR #4539 - The word "character" was misspelled in the instruction list at the start of CISREG. The problem has been corrected.

4. SR #5859 - CISREG did not list all classes for students enrolled in courses without meeting day or time specified. Problem corrected.

5. SR #4538 - CISREG did not differentiate between normal and maximum seat size when issuing message for filled classes. Problem corrected.

6. SR #5865 - CISBAR rejected only the first class when student registers in classes offered by a school in which he/she is not enrolled. All subsequent classes in that school were accepted. This problem has been corrected.

7. SR #5860 - CISRPT1 terminated without any error message but did not produce a class list when default values were specified for individual sections, card 3 was omitted, and the desired value for the day/night flag was entered in column 22 of card 2. This problem has been corrected.

8. SR #5187 - CGRADE was updating variable units field in log file when units were actually not variable. This problem has been corrected.

9. SR #5146 - CISBAT was not updating enrollment field in Course-Section data set when registration records were recovered from transaction log file. This problem has been corrected.

10. SR #4546 - CGRADE - Problem of CIS/3000 allowing students enrolled in one school to register in classes of another school but not allowing their grades to be reported was corrected by the correction of SMR #5440 in which a truncation problem with the SCHOOL-ID field was fixed in CISBAT.

11. SR # 4545 - When a student enrolled for classes in a future term, CIS updated L-Term fields to indicate this future term was the last one in which the student was registered. Most users wanted the L-Term fields to reflect the last term in which a student actually attended classes. The L-Term fields were thus inaccurate and have therefore been disabled.

    The fields L-Term-Grd-Gpa, L-Term-Grd-U-Att, L-Term=Grd-U-Comp, L-Term-Reg, L-Term-Reg-Units, L-Term-U-Grd, L-Term-Grd-Pt, and L-Term-Graded have been disabled and are no longer updated automatically by CIS/3000. Manual update is still possible using CISBAT (card 58) or CISDET (format 26). Also disabled is the Academ-Stat-Stud field based on L-Term-U-Comp-Stud and L-Term-Grd-Gpa-Stud fields in CGRADE, CISGRD, and CISBAT.

12. SR #5863 - CISBAR registers the first course for a student, all olthers on that card are truncated. This was found to be a documentation error in the manual and has been corrected. On page 7-5 of the CIS Reference Manual, figure 7-4, an extra column was added to the Course Information to allow for a pass/no pass option. Thus the Course Information extends from column 11 to 29 for the first course entered and from columns 30 to 48 for the second. A third course would begin in column 49. Similar changes were made to pages 7-6 and 7-7 of the manual, figures 7-5, 7-6, and 7-7. An extra column, column 29, has been added to allow for a pass/no pass option.

13. SR #6917 - CISRPT2 printed incorrect class levels for students. This problem has been corrected.

14. SR #5858 - CISRPT2 - Courses which have no meeting days and no times specified in course section detail are omitted when producing class schedules using CISRPT2. The total units are correct for these students. This problem has been corrected.

15. SR #9581 - The item list of DBGET specifies all items (@) rather than individual items. This was corrected in the COPYLIB.

C. Known Problems.

1. SR #4544 - In Format 20, an invalid status code results in a non-meaningful error message and no opportunity to correct the error.

2. SR #4547 - After changing edit information with Format 08, CISSCH will not allow user to specify and S or A Transaction Code.

3. SR #5263 - CISSCH FORMAT08 will not allow a user to change multiple edit in a table without entering a different transaction using a different format and table.

4. SR #5264 - CISCRS on FORMATS 33 and 39 the cursor does not return to the home position, and a second change for the same ID# will not work. If an add is done after the change for the same ID#, the add works as a change.

5. SR #5265 - CISCRS produces error messages that contain a portion of a previous error message.

6. SR #5266 - A user must have access to all Formats to update S, C, A, or D on Formats 07 and 08.

7. SR #5267 - Two and three deigit phone extensions do not pass the numeric edits.

8. SR #6009 - A student schedule prints the number of units for a variable unit class from the course master rather than from the student's record.

9. SR #6256 - CISGRD aborts and reports a positive image error number rather than a negative number, when the password given has insufficient capability for grading.

10. SR #6010 - Second page of a student's schedule is printed unnecessarily when the number of classes attempted is sufficient to fill a single schedule exactly.

11. SR #6965 - CISGRD will not accept grades for courses with zero credits.

12. SR #7022 - In CISGRD final grades cannot be entered if midterm grades have already been posted.

13. SR #7842 - User unable to change student status using CISBAT, card format 61. He supplied valid student ID and school ID but got invalid dept and major ID's in error message.

14. SR #7692 - When user only has access to Format 19, CISDET displays error message "Invalid Format Request for Cisdet" when Formatl9 requested.

15. SR #7250 - Section Fee field presently allows for only three digits. It should allow for four.

16. SMR #7136 - CISRPT2 failed to print student ID cards of schedules for 2 students registered for current term. CIS failed to change their status from "P" to "A" despite their being registered.

17. SR #7138 - In CISRPT2 a class which had been dropped was printed on a student schedule.

D. Miscellaneous.

1. The following types of files appear on the MAT:

a. SxxAnnnF -- source files for on-line maintenance
   SxxSnnnF -- source files for all other programs
b. MxxAnnnF -- maintenance files for on-line
   MxxMnnnF -- maintenance files for all other programs
c. JxxJnnnF -- job files to compile CIS programs
   JxxAnnnF -- job files to compile CIS on-line maintenance programs
d. UxxUnnnF -- USL files
   UxxAnnnF -- USL files for on-line maintenance programs
e. BxxBnnnF -- linkage file used in preparing when two or more files must be linked together
   BxxBnnnF -- linkage file for on-line maintenance programs

f. PxxPnnnF -- program files

where

xx = file identification number
nnn = last three digits of product number
F = product version level

New JCL has been written to compile all programs in CIS/3000:

CISBAT -- J05J902A-J51J902A, J60J902A
link file -- B02B902A
CISDET -- J01A902A-J48A902A, J58J902A
link file -- B03A902A
CISCRS -- J49A902A-J61A902A, J58J902A
link file -- B04A902A
CISSCH -- J62A902a-J69A902A, J58J902A
link file -- B05A902A
CISREG -- J52J902A, J56J902A, and J57J902A
link file -- B06B902A
CISBAR -- J53J902A
link file -- B07B902A
CISGRD -- J54J902A
link file -- B08B902A
CGRADE -- J55J902A, J58J902A, and J59J902A
link file -- B09B902A
CISRPT1 -- J59J902A, J61J902A, and J64J902A
link file -- B10B902A
CISRPT2 -- J59J902A, J62J902A, and J64J902A
link file -- B11B902A
CISRPT3 -- J59J902A, J63J902A, and J64J902A
link file -- B12B902A

2. L-Term fields have been disabled to correct SMR # 4545 and are no longer updated automatically by CIS/3000. Manual update is still possible using CISBAT (card 58) or CISDET (format 26).

3. Log file U00U902A's Support account name has been changed to U77U902A and its source (S77S902A) and its maintenance file (M77M902A) have been included.

EDC/3000 HP32380A.01.01
DATE CODE 1932, NOON380A.HP32380.SUPPORT

A. ENHANCEMENTS

None.

B. CORRECTIVE SOFTWARE CHANGES

The following problems have been corrected in EDC.

1. SR #7264 - The indented bill of material only picked up the first engineering change on parts with multiple engineering changes. This problem has been fixed in EDC2100P.EDCPGM.

2. SR #9074 - When a blank text field was entered as a remark for a parent-part, unpredictable text appeared in the EDC on-line retrieval of a single-level bill with remarks (for the parent-part). This problem has been fixed by modifying the severity code of the text field in TBLIN580.EDCWORK.

3. SR #9314 - EDC3000 did not perform a value-check for CUR-OVH-RATE although the value-check flag was set to "Y" in the edit tables. This problem has been fixed in subprogram EDCVAL called by EDC0100P, and EDC0800P.

4. SR #9611 - WCTR-ID was not updated in the ROUTING data set, after a valid transaction was entered on the EDCFMT40 screen, "CHANGE DESCRIPTIVE DATA". This problem has been fixed in EDC0590P.

5. SR #9869 - TBL0100 aborted with a bounds violation after having written out a new edit file. The outputted edit file contains all changes. This problem has been fixed by recompiling the program.

6. SR #10101 - The default values of CUR-TL-SETUP-HR, CUR-TL-RUN-HR, CUR-LL-SETUP-HR, and CUR-LL-RUN-HR were initialized to -.0001. These fields should have been initialized to zero. This problem has been fixed in EDC0370P.EDCPGM.

C. KNOWN PROBLEMS

1. SR #5715 - TBL0100 does not recognize MPE III command intrinsic errors.

D. MISCELLANEOUS

1. SL code segments were combined to reduce the maximum number of entries loaded in the Code Segment Table from 9 to 2.

IOS/3000 HP32384A.01.01
DATE CODE 1932, N00N384A.HP32384.SUPPORT

A. ENHANCEMENTS

1. SR# 9109, SR# 9702 - The Extra Usage Report was enhanced by the inclusion of part descriptions and configuration codes. This enhancement has been added to IOS/3000 by modifying IOS1800P.IOSPGM and IOS1850P.IOSPGM.

2. SR# 9695 - The Shortage/Pre-shortage Report was enhanced by the inclusion of part descriptions. This enhancement has been added to IOS/3000 by modifying IOS0900P.IOSPGM and IOS0950P.IOSPGM.

3. SR# 9703 - The Material Requisition was enhanced by the inclusion of parent-part descriptions. This enhancement has been added to IOS/3000 by modifying IOS0610P.IOSPGM, IOS0620P.IOSPGM, and IOS0630P.IOSPGM.

4. SR# 9704 - The IOS0400J/IOS0450J jobstream was enhanced by the inclusion of part descriptions for updated parts. This enhancement has been added to IOS/3000 by modifying IOS0450P.IOSPGM.

B. CORRECTIVE SOFTWARE CHANGES

All of the following known problems have been corrected in IOS.

1. SR# 5239 - The Inventory Report did not show a total inventory value for all controllers. This problem has been fixed in program files, IOS1400P.IOSPGM and IOS1450P.IOSPGM.

2. SR# 5241 - The Inventory Value Report should be sorted by controller, six month requirement, and then by part number. This problem has been fixed in JCL file, IOS1450J.IOSJCL.

3. SR# 5244 - The Purchase Committment Report truncated the report total after $9,999,999.99. This problem has been fixed in program file IOS1650P.IOSPGM.

4. SR# 5276 - IOS screen ORDFMT19 incorrectly infers that released orders can be deleted. This problem has been fixed by changing the forms file for that screen in ORDFORMS.IOSWORK, used by ORD0100P.IOSPGM.

5. SR# 5714 - IOS0720P aborted when the shop calendar was used and the pull date was outside the range of the shop calendar. This problem has been fixed.

6. SR# 5721 - The routine to partially fill backorders looped through the backorder chain only once rather than continuing to fill with partial quantities until all backorders were filled or until the quantity received was exhausted. This problem was fixed in the program file INV0100P.IOSPGM.

7. SR# 5952 - When the material requisition limit feature of the Material Requisition jobstream was used, IOS0610P did not continue to activate allocations after having reached this limit. This problem has been fixed in IOS0610P.IOSPGM.

8. SR# 7893 - IOS0720 created allocations for phantom parts with a status of inactive, allowing the same phantom parts to be allocated multiple times. This problem was fixed in the subprogram INVDB called by IOS0200P, IOS0610P, and IOS0720P.

9. SR# 9115 - When SHOP-CAL-INDEX was outside of the index table, GET-CAL-DATE would return an error-code of "00", indicating that all data was valid. This problem has been fixed in subprogram CAL1200 which is called by IOS0720P.IOSPGM.

10. SR# 9612 - Planned ("PL") and Planned Firm ("PF") orders did not appear on the Scheduled Receipts and Issues screen (INVRET01). This problem has been fixed in the subprogram INVRTRV, called by INV0100P.IOSPGM.

11. SR# 9769 - IOS0720 aborted with an error message of "too many phantoms" when more than ten parts existed in a structure chain, although the maximum is 50. This problem has been fixed in IOS0720P.IOSPGM.

12. SR# 9770 - IOS0710 aborted with a bounds violation whenever a work-order had more than ten phantom parts in a structure chain. This problem has been fixed in IOS0710P.IOSPGM and the called subprogram IOS0715.

C. KNOWN PROBLEMS

1. SR #5258 - The Inventory Control portion of IOS allows a user without ADD/DELETE capability to add and delete allocations.

2. SR #5274 - It is possible to create multiple backorder records for a single extra usage order number.

3. SR #5651 - When originator 95 is deleted, the error message displayed in the Allocation Maintenance Jobstream refers to originator 99.

4. SR #5718 - Entering all 9's in an Adjust On Hand transaction results in the inventory balance being adjusted by a "-1".

5. SR #6209 - When IOS0200 aborts there is no information to indiate how many records have been processed.

6. SR #8408 - Specifying "ND" as the Issue-Method for an extra-usage allocation results in that allocation appearing on a material requisition and pick list.

D. DOCUMENTATION CHANGES

Certain pages in the IOS/User's Reference Manual will be modified to include changes made by adding part descriptions to certain IOS/3000 reports. The following pages are affected:

Page 4-53: This page will show a sample of the newly modified Extra-Usage report.

Page 4-65: Figure 4-18 will present a modified Shortage/Pre-shortage report.

Page 4-91: Figure 4-23 will present a modified Material Requisition.

In addition, pages 4-26 and 4-27 will be modified to include changes to the IOS inventory retrieval screen, INVRET01.

Page 4-26: Figure 4-7 will show how a planned order is indicated on the retrieval screen.

Page 4-27: The text will be modified to include the planned order indicator.

E. MISCELLANEOUS

1. SL code segments were combined to reduce the maximum number of entries loaded in the Code Segment Table from 9 to 2.

MRP/3000 HP32388A.01.01
DATE CODE 1932, N00N388A.HP32388.SUPPORT

A. ENHANCEMENTS

1. SR #9259 - Part number pegging has been added to the MRP Action Report. The MRP/3000 programs, MRP1000P.MRPPGM through MRP3000P.MRPPGM, have been modified to include part number pegging. The change includes modifications to the independent supply and requirement file MRP1055D. See the documentation section below for more information.

B. CORRECTIVE SOFTWARE CHANGES

1. SR #6338 - MRP planning did not suggest an order number for a phantom part under a set of specific conditions. This problem has been fixed in programs MRP1000P, MRP2000P, and the called module, MRP2020.

2. SR #7068 - In some cases, MRP2020 did not correctly calculate the order quantity for the order point order policy when it suggested the first order. This problem has been fixed in the called module, MRP2020.

3. SR #7142 - When the order point order policy is used, MRP did not suggest an order when the quantity available was equal to the order point. This problem has been fixed in the called module, MRP2020.

4. SR #9115 - When SHOP-CAL-INDEX was outside of the index table, the called module, GET-CAL-DATE would not return an error code, rather it indicated that all data was valid. This problem has been fixed in subprogram CAL1200U.UTILITY called by MRP1000P, MRP1050P, MRP2100P, MRP2500P, and MRP3000P.

5. SR #10366 - The JCL file, MRP2007J.MRPJCL of the MRP planning job stream, had an incorrect file equation. The file equation, !FILE RQMT = RQMT07.MRPWORK,OLD should have read, !FILE RQMT = RQMT.MRPWORK,OLD. This problem has been fixed in MRP2007J.MRPJCL.

C. KNOWN PROBLEMS

None.

D. DOCUMENTATION CHANGES

Certain pages in the MRP/3000 User/Administrator's Manual will be modified to include changes made from the addition of part-number peggin to MRP/3000. The following pages are affected.

Pages vi, vii, 1-1, and 1-2: These pages will include SPC/3000 as one of the MFG/3000 products.

Pages 2-14, 2-15, and 3-4: These pages will include samples of the newly modified Action Report.

Page 3-8: The text will be modified to include RQMT-PART-NUMBER, (the part-number which triggered the order).

Pages 8-2 and 8-3: These pages will display the new file layouts for the input file MRP1055D, which are:

```
01  INDEPENDENT-SUPPLY .

    05 CTLR                  PIC  9(02).
    05 PART-NUMBER           PIC  X(18).
    05 INDX-DATE-DUE         PIC S9(04)     COMP.
    05 RECORD-TYPE           PIC  X(01).
       88 SUPPLY-RECORD                              VALUE
       "2".
    05 ORDER-NUMBER          PIC  X(08).
    05 ITEM-NUMBER           PIC  X(02).
    05 MRP-ACTION            PIC  X(01).
       88 SUPPLY-DATE-UNCHANGED                      VALUE
       "0".
       88 SUPPLY-FIRM                                VALUE
       "1".
       88 SUPPLY-PUSHED-OUT                          VALUE
       "2".
       88 SUPPLY-PULLED-UP                           VALUE
       "3".
       88 SUPPLY-CANCELED                            VALUE
       "4".
       88 SUPPLY-WI-WINDOW                           VALUE
       "5".
    05 MRP-ACTION-9          REDEFINES  MRP-ACTION
                             PIC  9(01).
    05 SUPPLY-TYPE           PIC  X(01).
       88 RUN-SUPPLY                                 VALUE
       "0".
       88 WORK-ORDER-SUPPLY                          VALUE
       "1".
       88 PURCHASE-ORDER-SUPPLY                      VALUE
       "2".
    05 SUPPLY-STATUS         PIC  X(01).
       88 OPENED-ORDER                               VALUE
       "1".
```

```
      88 PLANNED-ORDER                          VALUE
      "2".
      88 SUGGESTED-ORDER                        VALUE
      "3".
   05 SUPPLY-STATUS-9     REDEFINES SUPPLY-STATUS
                          PIC  9(01).
   05 SUPPLY-QTY          PIC S9(07)     COMP-3.
   05 PREV-INDX-DATE-DUE  PIC S9(04).    COMP.
   05 VENDOR              PIC  X(08).
   05 GEN-COMPONENT-RQMT-FLAG      PIC X(01).
      88 GENERATE-RQMT-FOR COMPONENT          VALUE
      "Y".
      88 DO-NOT-GEN-RQMT-FOR-COMP             VALUE
      "N".
   05 SCHEDULING-EXCEPTION-FLAG    PIC X(01).
      88 NORMAL-SCHEDULING                    VALUE
      "0".
      88 FIRM-ORDER-EXCEPTION                 VALUE
      "1".
      88 IMPACTED-BY-FIRM-ORDER               VALUE
      "2".
   05 FILLER               PIC  X(06).

01 INDEPENDENT-RQMT.

   05 CTLR                 PIC  9(02).
   05 PART-NUMBER          PIC  X(18).
   05 INDX-DATE-DUE        PIC S9(04)     COMP.
   05 RECORD-TYPE          PIC  X(01).
      88 RQMT-RECORD                          VALUE
      "1".
   05 ORDER-NUMBER         PIC  X(08).
   05 ITEM-NUMBER          PIC  X(02).
   05 RQMT-STATUS          PIC  X(01).
      88 RQMT-DATE-UNCHANGED                  VALUE
      "0".
      88 PARENTS-SUP-FIRM                     VALUE
      "1".
      88 PARENTS-SUP-PUSHED-OUT               VALUE
      "2".
      88 PARENTS-SUP-PULLED-UP                VALUE
      "3".
      88 PARENTS-SUP-CANCELED                 VALUE
      "4".
      88 PARENTS-SUP-WI-WINDOW                VALUE
      "5".
   05 RQMT-STATUS-9    REDEFINES RQMT-STATUS
                       PIC  9(01).
   05 RQMT-TYPE         PIC  X(01).
      88 ALLOCATION-RQMT                      VALUE
      "2".
      88 BACKORDER-RQMT                       VALUE
      "3".
      88 EXTRA-USAGE-RQMT                     VALUE
      "4".
```

```
   88 EXPLODED-RQMT                          VALUE
   "5".
05 RQMT-TYPE-9          REDEFINES RQMT-TYPE
                        PIC  9(01).
05 FILLER               PIC  X(01).
05 RQMT-QTY             PIC S9(07)    COMP-3.
05 RQMT-PART-NUMBER     PIC  X(18).
```

E. MISCELLANEOUS

1. SL code segments were combined to reduce the maximum number of entries loaded in the Code Segment Table from 9 to 2.

SPC/3000 HP32392A.00.01
DATE CODE 1932, N00N392A.HP32392.SUPPORT

A. ENHANCEMENTS

None.

B. CORRECTIVE SOFTWARE CHANGES

The following problems have been corrected in SPC.

1. SR# 9073 - The selection of a workcenter range in on-line SPC/3000 resulted in an empty file of parts to be selected for cost roll-up. This problem has been fixed in SPC2000P.SPCPGM.

C. KNOWN PROBLEMS

None.

D. MISCELLANEOUS

1. SL code segments were combined to reduce the maximum number of entries loaded in the Code Segment Table from 9 to 2.

IMAGE Database Schema Survey

A survey of how customers design their IMAGE databases.

By John Page, General Systems Division

Here in the Database Management part of the GSD lab, we have a need for some data relating to the ways customers are designing their IMAGE databases. To help us make certain design decisions in our future products, we need to know things like:

(a) How many datasets in a typical database?
(b) How big are the datasets?
(c) What is the record length size distribution?
(d) Which are the most popular datatypes?
.
.
etc.

Your help in this effort would be appreciated. What we need is a mag tape with your schemas stored on it. We will then analyze all the schemas we receive and compile the statistics we need.

If you would like to contribute to this survey, here's what to do:

(a) Make a :STORE tape containing all your schemas in source form. Any density will do (1600 preferred) and you can use any account/group/creator since we will use the RESTORE program to retrieve the files.

(b) Mail the tape to:

Mike Huey Bldg. 48N
Hewlett-Packard Co.
General Systems Division
19447 Pruneridge Ave.
Cupertino CA 95014

We would like only schemas for databases that are actually in operational use, since experimental and test databases are often unrealistically small and this would pollute our dataset size statistics.

If you are interested in the results, make a note on the tape and if there is sufficient interest, I will publish them in the Users Group Journal.

I hope you can help us. The data we receive will help us build better systems for you in the future.

IMAGE: Address Calculation; QUERY: Reporting Techniques

This article contains a brief discussion of IMAGE's address calculation algorithms followed by notes on generating reports in QUERY by using REGISTER statements and nested XEQ files.

by Tony Lemberger, General Systems Division

IMAGE: A Word on Address Calculation Algorithms and Efficiency

There are two address calculations -- one for byte data (IMAGE type X,U, or Z) and one for word data (IMAGE type I,U, or R). As usual, there are benefits and costs associated with each.

Byte Data -- A hashing algorithm is applied to the byte data yielding a double integer, then modulo arithmetic is used to produce a relative record address.

Word Data -- Modulo arithmetic is used to produce a relative record address.

The choice of data type can affect performance. Unless the user is doing his own randomizing of key values, byte data types are best. While the initial overhead is slightly higher for byte data (the hashing algorithm), a vast improvement in response time is achieved due to the even spread of data in the data base files. If user keys are sequential or grouped, use byte data types!

QUERY: Notes on Register Arithmetic

Some things to remember when writing QUERY REPORT Procedures that use register arithmetic:

1) All register statements are done before any detail statements.

2) All register statements are executed for each record processed by a report procedure.

3) Register statements are executed in the order they appear in the REPORT Procedure, regardless of any intervening statements.

Two examples should illustrate these points. First, this report

```
REPORT
R1, LOAD, "1000"
D1, R1, 10
R1, LOAD, "2000"
D1, R1, 20
END
```

generates "2000 2000", not "1000 2000". Because all register statements are executed before any detail statements, the above report loads register one with "1000", then again with "2000" (replacing the "1000"), and then uses the detail statements to produce a line of output.

As a second example, consider these two reports -- both written to find the cumulative difference between "scheduled hours" and "actual hours" for each Job-ID. The totals generated do not always match; the one on the left is sometimes too large.

```
REPORT                          REPORT
S1, JOB-ID                      S1, JOB-ID
R1, ADD, SCHED-REG-HRS          R1, ADD, SCHED-REG-HRS
R1, ADD, SCHED-OT-HRS           R1, ADD, SCHED-OT-HRS
R2, ADD, REAL-REG-HRS           R2, ADD, REAL-REG-HRS
R2, ADD, REAL-OT-HRS            R2, ADD, REAL-OT-HRS
T1, R1, 10                      T1, R1, 10
T1, R2, 20                      T1, R2, 20
R3, ADD, R1                     R3, LOAD, R1
R3, SUBTRACT, R2                R3, SUBTRACT, R2
T1, R3, 30                      T1, R3, 30
T1, R1; T1, R2; T1, R3          T1, R1; T1, R2; T1, R3
END                             END
```

The reason the totals do not match is that all register statements are executed for every record processed. Note that on the left, register one is always added to register three before register two is subtracted. But register three already contains the difference of any records for that Job-ID (if any records have been previously processed). For 5 records processed, the difference for the first record is included in register three 5 times!

Generating Reports from Multiple Data-Sets in QUERY

QUERY does not allow Report Procedures that reference more than one data-set. However, by using nested XEQ files, it is possible to produce reports with data from multiple data-sets.

The most straight-forward data-set combinations to work with are either two Masters or one Master with a related Detail. The basic technique is

1) A user written Report procedure is formatted so that it creates an XEQ file containing data from the first data-set that information was drawn from, as well as a FIND statement that will select data from the next data-set.

2) The XEQ file created in step one is then executed. If only two data-sets are being used, the output it creates is the desired report. If more than two data-sets are being used, then the output is formatted as an XEQ file. The process can be nested indefinitely (because XEQ files can be nested indefinitely), but the last XEQ file to be executed produces the desired report.

The two examples that follow are based on the STORE data base as described in the IMAGE Reference Manual.

For ease of presentation, the first example will begin with the desired report and work backward. (This is also the safest way to design this kind of XEQ file!) The first report to be produced is--

```
NAME FROM MASTER          DATE FROM DETAIL
  SCHLILLIGAN                  JUN 76
  SCHLILLIGAN                  MAY 78
```

(Simple maybe -- but a more complicated one follows.)

An XEQ file capable of producing such a report might appear as follows.

```
FIND SALES.ACCOUNT=      1 <---note the extra spaces
REPORT
H1,"NAME FROM MASTER",16
H1,"DATE FROM DETAIL",40
D1,"SCHLILLIGAN     ",18 <---this is the LAST-NAME field
D1,PURCH-DATE,35             from the CUSTOMER data-set!
END
```

Note that some of the data (specifically the LAST-NAME) appears as a literal. Note also those extra spaces in the ACCOUNT value and the LAST-NAME literal -- these lines were built by an XEQ file, and room had to be left for the largest possible data-item value. These literals are being used to pass variable data from XEQ file to XEQ file. In each XEQ file, a new data-set is being referenced, and the information garnered from each must be kept in literals until it is output in a report.

Such an XEQ file can be generated by the following XEQ file.

```
FIND CUSTOMER.ACCOUNT=1
REPORT NOPAGE LP
H1,     "FIND SALES.ACCOUNT="                    ,19
H1,     ACCOUNT                                  ,26
H2,     "REPORT"                                 ,6
H3,     "H1,""NAME FROM MASTER"",16"             ,24
H4,     "H1,""DATE FROM DETAIL"",40"             ,24
D1,     "D1,"""                                  ,4
D1,     LAST-NAME                                ,20
D1,     """,18"                                  ,24
D2,     "D1,PURCH-DATE,35"                       ,16
D3,     "END"                                    ,3
END
```

There are several things to note in this XEQ file. First, remember that in QUERY double quotes inside of quotes mean one quote embedded in a character string. Next note how the two Header One statements build the one FIND statement in the "output" listed above. And how the three Detail One statements build the Detail statement above that contains the literal. Also be aware of just how "messy" a simple report like this one became.

This second XEQ file would be the one the user would write. It, in turn, would create the first XEQ file, which would produce the desired report. If in the user-written XEQ file, the first line was FIND ALL CUSTOMER.ACCOUNT, then a much larger XEQ file would have been produced by it. In that "second-generation" XEQ file there would be a FIND and REPORT statement for each ACCOUNT in the CUSTOMER data-set. That series of FIND-REPORT, FIND-REPORT statements would produce a series of small related reports similar to the one we started with. By specifying NOPAGE when the report is produced, the user can avoid getting separate pages for each of these pieces of the total report.

Before the next example (which shows some more bells and whistles), the actual production of the above report is outlined.

After creating the initial XEQ in the EDITOR, the standard output file for QUERY must be directed to a disc file.

```
:BUILD HOLD;REC=-80,16,F,ASCII
:FILE QSLIST=HOLD,OLD;DEV=DISC
```

Then get into QUERY and execute your EDITOR file. (For XEQ files that are nested more than one deep, make the last statement of all but the last XEQ file be "XEQ HOLD", and, like dominoes, each piece will start the next piece!) When the final XEQ file has been built--the one that will produce your report--the file equation for QSLIST must be RESET. (This will direct your final output to the line-printer instead of a disc-file.) Then XEQ HOLD for the last time, and your multi-data-set report will appear on the line-printer.

This technique can be expanded to include entire applications. Data entry (with audit trails), report generation, and impromptu inquiry--all available, with QUERY. However, simple report generation can become quite complicated when QUERY is the base of the application.

However, nested XEQ files can be used to produce those occasional reports that are always needed which had to be produced with a quick and dirty program. The development time for nested XEQ files can be substantially lower than that for a program, once you get the hang of it.

Just a word more about other combinations of data-sets. When the user cannot predict how many records will be found, it is hard to nest XEQ files from that data-set to another. So combining several Details, or selecting data from a Master based on something other than the Key value is not as straight-forward. One possible implementation is to construct multi-level Detail statements on the run. Write a "D", followed by a register (incremented for every record processed--01,02,etc), followed by the data to be printed. This allows up to nine records to be processed. There is nothing to prevent a user from constructing group and total statements in the same manner, although the logic becomes even more complex.

EXAMPLE 2

If you are interested in learning more about this technique, the following example is more challenging. It shows a more complicated (and realistic) report. The user-written XEQ file is listed first, with its associated REPORT Procedure. Then the XEQ file it creates; and last the report that is generated.

```
    FIND ALL CUSTOMER.ACCOUNT
    REPORT SUMMARY

PROCEDURE: SUMMARY

REPORT NOPAGE LP
  H1,  "LP"                                  ,2
  D1,  "FIND SALES.ACCOUNT="                 ,19
  D1,  ACCOUNT                               ,26
  D2,  "REPORT NOPAGE"                       ,13
  D3,  "H1,""ACCOUNT"",10,SPACE B3"          ,24
  D3,  ";H1,""NAME"",16"                     ,37
  D3,  ";H1,""CREDIT-RATING"",58"            ,59
  D4,  "H2,"""                               ,4
  D4,  ACCOUNT                               ,10
  D4,  """,10;H2,"""                         ,19
  D4,  LAST-NAME                             ,35
  D4,  """,28;H2,"","",29;H2,"""             ,54
  D4,  FIRST-NAME                            ,64
  D4,  """,40"                               ,68
  D5,  "H2,"""",4;D5,CREDIT-RATING           ,14
```

```
  D5,  """,55,SPACE A2"                              ,27
  D5,  ";H3,""SALES HISTORY          STOCK#"",54"          ,60
  D6,  "H3,""QUANTITY PRICE  PURCH-DATE  DELIV-DATE"",100",52
  D7,  "D1,STOCK#,54;D1,QUANTITY,64;D1,   TOTAL,75"       ,39
  D8,  "D1,PURCH-DATE,86;D1,DELIV-DATE,  98"              ,33
  D9,  "S2,PURCH-DATE;S1,STOCK#;END"                      ,27
  TF,  "TERM"                                             ,4
END


LP
FIND SALES.ACCOUNT=      1
REPORT NOPAGE
H1,"ACCOUNT",10,SPACE B3;H1,"NAME",16;B1,"CREDIT-RATING",58
H2,"     1",10;H2,"JONES           ",28;H2,",",29;H2,"MIKE   ",40
H2,"   9.87700",55,SPACE A2;H3,"SALES HISTORY     STOCK#",54
H3,"QUANTITY     PRICE   PURCH-DATE  DELIV-DATE",100
D1,STOCK#,54;D1,QUANTITY,64;D1,TOTAL,75
D1,PURCH-DATE,86;D1,DELIV-DATE,98
S2,PURCH-DATE;S1,STOCK#;END
FIND SALES.ACCOUNT=      0
REPORT NOPAGE
H1,"ACCOUNT",10,SPACE B3;H1,"NAME",16;H1,"CREDIT-RATING",58
H2,"     0",10;H2,"SMITH           ",28;H2,",",29;H2,"FRED   ",40
H2,"   6.55556",55,SPACE A2;H3,"SALES HISTORY     STOCK#",54
H3,"QUANTITY     PRICE   PURCH-DATE  DELIV-DATE",100
D1,STOCK#,54;D1,QUANTITY,64;D1,TOTAL,75
D1,PURCH-DATE,86;D1,DELIV-DATE,98
S2,PURCH-DATE;S1,STOCK#;END
TERM


ACCOUNT NAME          CREDIT-RATING
  1 JONES      ,MIKE    9.87700
                SALES                                    PURCH-  DELIV-
                HISTORY  STOCK#   QUANTITY  PRICE  DATE    DATE
                         444444          1   1500  JUN 76  JUL 76
                         11111111       37  20965  MAY 78  AUG 78


ACCOUNT NAME          CREDIT-RATING
  0 SMITH      ,FRED     6.55556
                SALES                                    PURCH-  DELIV-
                HISTORY  STOCK#   QUANTITY  PRICE  DATE    DATE
                         11111111        7    900  OCT 78  JAN 79
                         33333333        1    920  OCT 78  OCT 78
                         22222222      100   1500  SEP 77  DEC 78
```

****
* Editor's Note: PURCH-DATE and DELIV-DATE should be on *
* one line; this section is compressed due to the *
* page-size limit of this magazine. *
****

Using HP VIEW/3000 VGETtype Intrinsics

Correlating HP VIEW/3000 data types, VGETtype intrinsics, and programming language data types.

by Carla Klein, General Systems Division

When using HP VIEW/3000 for data entry, all data is read from the unprotected fields on the screen as ASCII characters and concatenated to form a data buffer. The VGETBUFFER intrinsic will transfer the entire buffer to an application program or VGETFIELD can be used to obtain the contents of an individual field. However, neither of these intrinsics performs any conversion of the data values. Therefore, a field defined as DIG in FORMSPEC and containing the value 123 would be transferred as the string of characters "123". The VGETtype intrinsics, VGETINT, VGETDINT, VGETREAL, and VGETLONG, have been provided to perform conversion from ASCII to the four indicated data types. The following chart and notes will help you correlate VIEW/3000 data types, VGETtype intrinsics, and programming language data types.

LANGUAGE	Intrinsic	Data Type
FORTRAN	VGETINT	INTEGER
	VGETDINT	INTEGER*4
	VGETREAL	REAL
	VGETLONG	DOUBLE PRECISION
BASIC	VGETINT	INTEGER
	VGETREAL	REAL
	VGETLONG	LONG
SPL	VGETINT	INTEGER
	VGETDINT	DOUBLE INTEGER
	VGETREAL	REAL
	VGETLONG	LONG
COBOL	VGETINT	S9-S9(4) COMP
	VGETDINT	S9(5)-S9(9) COMP

***** ADDITIONAL NOTES *****

1. If errors occur during conversion, the status word in the communications area is set to an error value.

2. If the requested field has been flagged as having an error (perhaps by VFIELDEDITS or VSETERROR), the conversion is performed, but the status word is also set to an error value.

3. An attempt to convert a number larger than 32767 using VGETINT will return an error value (504) in the status word and will leave the receiving value unchanged.

4. All commas are stripped before conversion is performed.

5. Fields of type CHAR may be converted as long as they contain numeric characters (including " . , - + ") Otherwise an error value will be returned in the status word.

6. VGETINT and VGETDINT will only convert the integer portion of a given field. The fractional portion is truncated before conversion. Remember that in a field of type IMPn, the rightmost "n" characters will be treated as a fraction.

7. If VGETFIELD is used to pass a field containing a decimal point to a COBOL program, the decimal point will also be passed and no arithmetic may be performed on the field.

8. Negative numbers can be zoned correctly for COBOL only by using the VGETINT and VGETDINT intrinsics. VGETBUFFER and VGETFIELD will transfer the negative sign, but COBOL will treat the value as positive, ignoring the sign character. An EXAMINE statement using TALLY can determine that the negative sign is present and then the program can treat the value accordingly.

9. Normal rules of truncation in COBOL are followed. For example, conversion of 12345 using VGETINT with a receiving field of S9(4) will truncate the value to 2345.

10. VGETINT may be used to convert positive integers to type LOGICAL in SPL.

HP VIEW/3000 User Education

by Babs Brownyard, General Systems Division

The HP VIEW/3000 user course is currently available at HP training facilities (HP22830A) or as on-site instruction (HP22830X). The class is scheduled for four days. The first two days focus on screen design and the use of FORMSPEC, ENTRY, and the reformatting capability. This portion of the course does not require programming knowledge on the part of the student.

The second segment is devoted to programmatic use of HP VIEW/3000; that is, the use of the HP VIEW procedures. Lecture material concentrates on the interaction of the procedures, while labs guide the students in the actual use of application programs. The entire course is heavily lab-oriented, emphasizing hands-on use of the product.

HP VIEW features such as menu selection and extensive documentation enable most users to design basic screens without instruction. The course enables the user to move beyond the fundamentals of the product and to employ its many advanced capabilities. Students are in a much better position to use it effectively and efficiently in their programs.

Contact your HP Sales Representative for more information on the HP VIEW/3000 User Course, or for information on instructional materials for your own in-house training program.

Analysis of RPG Run-time Error Aborts

A procedure to relate RPG run-time error aborts, caused by user program code, to a source code line number.

by Geoff Walker
HP-Paramus, New Jersey

I. RPG Feature

The RPG compiler loads and stores the line number for each statement it executes, so that when a run-time error appears RPG can inform the user where the error occurred. If this feature is not needed in a program the user can save up to 30% of code space (resulting in faster execution, especially of large programs) by entering an N in column 20 of the header record.

II. Procedure

1. Assume you get a run-time bounds violation such as the following:

```
ABORT :PROGRAM.GROUP.ACCOUNT.%1.%3044
PROGRAM ERROR #24 :BOUNDS VIOLATION
*** ABORT STACK ANALYSIS ***

S=011341      DL=177602      Z=014162
Q=011345 P=003044  LCST=  001 STAT=U,1,1,L,0,0,CCG X=000000
Q=011336 P=004747  LCST=  001 STAT=U,1,1,L,0,0,CCG X=000001
Q=011331 P=002270  LCST= S137 STAT=U,1,1,L,0,0,CCL X=000034
Q=011312 P=000022  LCST=  000 STAT=U,1,1,L,0,0,CCG X=000000
Q=011306 P=177777  LCST= S055 STAT=P,1,0,L,0,0,CCG X=000000
```

   NOTE: The "ABORT STACK ANALYSIS" information is produced by entering the MPE command ":SETDUMP" and running the program in question.

2. From the above information, the RPG program is aborting in segment (LCST) #1, at P-location 003044. The absence of an "S" in front of the code segment number indicates the abort is occurring in user code, not system SL code (as is sometimes the case).

3. The first step is to run "DECOMP" on the program file, specifying the number of the segment you want to decompile (#1 in this case). This will produce a listing of the machine instructions for the segment.

4. Find the abort P-location in the left hand column of the DECOMP listing (3044 in this case).

5. Read backwards from that point (up the page) until you find the following combination of instructions:

```
LOAD  P+/- ###        (### > 255)
STOR  DB+064

OR...

LDI   ###             (### < 256)
STOR DB+064
```

6. Location DB+64 is used to store the number of the current source code line being executed. For source line numbers above 255, "###" is a displacement in the code pointing to the storage location containing the line number. For source line numbers below 256, "###" itself is the source line number (in octal). For example, suppose the load instruction is at location 3030 (the number in the left-hand column) and it reads "load P-41". You must subtract 41 from 3030 (in octal), giving 2767, the location containing the current source line number.

7. Now look up location 2767 in the left-hand column of the DECOMP listing. The value in the second colmn is the current source code line number *IN OCTAL*. Note that DECOMP will incorrectly interpret this value as one or two machine instructions instead of a constant. Translate the value from octal to decimal, and you're done!!

FOOTNOTES:

1. The "calculator" mode of DEBUG can be used to simplify the above octal arithmetic and conversions. An example is as follows (see the DEBUG manual for further information):

```
:EDITOR
/DEBUG
?=3030-41
=2767
?=1263,I
=+691
?E
END OF SUBSYSTEM
```

2. In the above abort example, the actual abort location was in a subroutine entered from several different places in the program. The stack trace-back can be used to tell where the subroutine was called from: following the above procedure for location 4747 in the same code segment (from the second line of the abort stack analysis) will produce the line number from which the subroutine was called.

3. If you're not sure you've done the above procedure correctly, try it again, this time going backwards to the load/store pair before the one you just examined. Chances are this pair will point to the previous source code line number.

4. Note that the above procedure cannot be used if the line number suppression feature (an "N" in column 20 of the control card) has been invoked.

Iterative Operations with EDIT/3000

Use of the FLAG in WHILE loops.

By Jerry Schwartz, HP - Rockville, Maryland

One of the most powerful facilities of EDIT/3000 is the WHILE command, which allows repetitive operations. In order to use it to the utmost, especially in situations where conditional execution of certain commands is necessary, understanding of a little-known facility called the FLAG is invaluable.

The FLAG reflects the success or failure of an EDITOR command, and is most useful in WHILE loops, where it may be referenced explicitly or implicitly.

To understand the use of the FLAG, it must be remembered that the execution of a WHILE command has the structure:

```
WHILE <condition> do <operation>
```

Where <operation> is an EDITOR command (or block of commands) and <condition> has one of the values "yes" or "no", corresponding to the success or failure of the immediately preceding EDITOR command, respectively. In its simplest form, <condition> will be the FLAG. For example,

```
/WHILE FLAG
/   LIST "FRED"
```

will list all lines containing the string "FRED" from the current pointer position to the end of file (subject to the iteration limit set by the SET TIMES = command). This, by the way, is about the fastest way of listing all occurrences of a string.

The command LIST <string> will behave as follows:

1. Search forward from the current pointer position for an occurrence of <string>. If not found, set FLAG to "no" and stop with a message.

2. List the line found.

3. Advance the pointer to the start of the next line; or (if the line just listed was the last) leave it at the end of the last line.

The LIST command is executed repeatedly until FLAG is set to "no" by failure of (1) above. Because LIST advances the pointer to the next line (3), a line will never be listed more than once. Compare this with

```
/WHILE FLAG
/   FIND "FRED"
```

Unlike the LIST command, FIND leaves the pointer at the beginning of the string. This loop would display the same line until the number of iterations exceeded TIMES.

The examples described above are pure in the sense that FLAG is used as the <condition>. FLAG has no side effects. Often, however, a command (or block of commands) is used as the <condition>. Most commands have side effects; they change the text or move the pointer in addition to having a resulting value of "yes" or "no". This makes it imperative that close attention by paid to where each command leaves the pointer. For example,

```
/WHILE
/   LIST "FRED"
/   DELETE *
```

will list each line containing the string "FRED" and then delete the line immediately following. The similar-looking construct

```
/WHILE
/   FIND "FRED"
/   DELETE *
```

will delete the lines containing "FRED", as would the more straightforward

```
/WHILE FLAG
/   DELETE "FRED"
```

The WHILE command always evaluates the <condition> regardless of the value of the FLAG. In other words, if the <condition> consists of one or more commands then at least the first will be executed.

Multiple commands may be used in either <condition> or <operation> if they are bracketed by BEGIN-END pairs. In the <condition>, this has the effect of logically ANDing the commands together. Since if the first of two or more logical values being ANDed together is "no" (true) the others need not be evaluated, once the FLAG is set to "no", then subsequent commands in <condition> are ignored. Similarly, if multiple commands are blocked together in <operation>, once the FLAG is set to "no", then subsequent commands are ignored. Consider the following construction:

```
/WHILE
/   BEGIN
/      FIND "LUMPY"
/      FIND * (FIRST)
/      FIND "ROTTEN" / * (LAST)
/      END
/   CHANGE "POTATOES" TO "SPUDS" IN *
```

Three FIND commands are needed to locate all lines in which both "LUMPY" and "ROTTEN" occur, regardless of the order in which they appear. The first FIND locates a line containing "LUMPY"; the second (which can never fail) repositions the pointer to the beginning of that line; and the third locates the string "ROTTEN" if it occurs before the end of the line. If any FIND fails, FLAG is set to "no", the remainder of the <condition> block is skipped, and the WHILE terminates. Similarly, consider

```
/WHILE
/   FIND "LUMPY"
/   BEGIN
/      DELETE * (+5) / * (LAST)
/      FIND * (FIRST)
/      FIND "ROTTEN" / * (LAST)
/      CHANGE "LUMPY" TO "" IN *
/      END
```

This will locate an occurrence of "LUMPY", then delete everything after "LUMPY" to the end of the line. The pointer is then moved to the beginning of the line, and a search made for "ROTTEN" on that same line. If "ROTTEN" is found, "LUMPY" is changed to a null string; if "ROTTEN" is not found, the CHANGE command is ignored. The loop continues with evaluation of the <condition> (FIND "LUMPY") in either case. Unfortunately, the above sample contains an error: if "ROTTEN" is not found, the pointer is left at the beginning of the line, and the next iteration of the loop will find the same occurrence of "LUMPY". In fact, the previous example contains a similar bug: if the string "POTATOES" precedes "LUMPY" on the same line, the pointer will never advance to the next line, and the EDITOR will loop until the iteration limit is reached.

The EDITOR gives us some commands to perform additional logical operations: NOT, OR, and YES. These commands will always be executed when encountered, regardless of the value of the FLAG. To fix our last example, we need to use the OR command, which performs an inclusive OR operation with a side effect: since a "yes" (true) result from the first of two conditins being ORed together obviates the need to evaluate the second, the command immediately following an OR command is executed (evaluated) only if the FLAG is "no" (false) when the OR is encountered. Returning to our example,

```
/WHILE
/   FIND "LUMPY"
/   BEGIN
/      DELETE * (+5) / * (LAST)
/      FIND * (FIRST)
/      FIND "ROTTEN" / * (LAST)
/      CHANGE "LUMPY" TO "" IN *
/      OR
/      FIND * +1
/      END
```

The addition of the OR and FIND * +1 has this effect: if the FIND "ROTTEN" / * (LAST) succeeds, the CHANGE command is executed, but the FIND * +1 is not; or, if the FIND "ROTTEN" / * (LAST) fails (setting the FLAG to "no"), then the CHANGE command is skipped, the OR sets the FLAG to "yes", and the FIND * +1 is executed. Bear in mind that if the statement following an OR is executed but fails (sets the FLAG to "no"), the FLAG remains set to "no". Use of an OR does not guarantee the value of the FLAG; it is a means for conditionally executing a single command or command block.

The other two logical operators, NOT and YES, are also useful. YES will simply set the FLAG to "yes" whenever it is encountered. For example,

```
/WHILE
/   FIND "LUMPY"
/   BEGIN
/      FIND "MATTRESS" / * (LAST)
/      LIST * -1
/      YES
/      LIST *
/      END
```

will find each occurrence of "LUMPY" and, if the string "MATTRESS" follows it on the same line, the line before is listed; in any event, the line originally found is listed. (Remember what the LIST command does to the pointer.)

Last but not least, the NOT command reverses the sense of the following command.

```
/WHILE
/   FIND "LUMPY"
/   BEGIN
/      NOT
/      FIND "MATTRESS" / * (LAST)
/      LIST *
/      END
```

This WHILE loop will list all lines,on which "LUMPY" is found but is not followed by "MATTRESS".

Two final notes are in order. Normally, the EDITOR lists each command in a loop as it is encountered; if the command was skipped, a dash ("-") appears to the left. This is useful for debugging but messy for general use. The SET SHORT command will suppress this listing. Judicious use of the Q suffix (e.g., FINDQ) will improve the appearance of listings still further.

Finally, the FLAG is also usable in command files invoked with the USE command. The same rules apply as would apply to the <operation> part of a WHILE loop. No looping occurs unless another USE command with no parameters will return to the previous command file, and may be used with a FLAG and OR, NOT, and YES commands to conditionally execute parts of a command file.

As an example, consider the following USE file:


```
Q "What is the name of your file?" Z ::=
TEXT  Z::
OR
USE                                  <<Return if TEXT failed>>
FINDQ "$CONTROL" / FIRST (LAST)      <<Start with $CONTROL?>>
FIND *+ +1                           <<If yes, skip line>>
YES                                  <<In either case, continue>>
FIND  "IDENTIFICATION DIVISION." / * (LAST)
OR                                   <<If not found, stop>>
BEGIN
   Q "File does not begin with IDENTIFICATION DIVISION"
   USE                               <<return>>
   END
...
```

This command file performs a certain amount of checking before continuing with whatever work it has to do. A file must be successfully TEXTed; and it must begin with "IDENTIFICATION DIVISION", or with "$CONTROL" immediately followed by "IDENTIFICATION DIVISION". If these conditions are not met, it stops executing and returns control to the user.

PARM=Values When Running a Compiler Program

This article defines the range of values which may be used with the PARM= parameter when HP 3000 Compilers are invoked via the MPE RUN command.

By John Pavone, General Systems Division

The HP 3000 MPE Operating System allows the user to invoke compiler subsystems in two ways.

Method 1 is via the MPE command :Compiler-name followed by the optional parameter list which declares the ACTUAL FILE DESIGNATORS that the compiler is to use for the Text, Usl, List, Master & New files.

Method 2 is via the MPE command :RUN Progfile and passing a Equated-files-value via the PARM= parameter. The Equated-files-value informs the compiler which of the ACTUAL FILE DESIGNATORS have been equated to the compilers FORMAL FILE DESIGNATORS via File Equations.

The following tables identify the FORMAL FILE DESIGNATORS for each compiler subsystem and the range of valid PARM= values and their associated File Equation declaration combinations.

Table-1

Compiler	Formal File Designators				
SPL	SPLTEXT	SPLUSL	SPLLIST	SPLMAST	SPLNEW
FORTRAN	FTNTEXT	FTNUSL	FTNLIST	FTNMAST	FTNNEW
COBOL	COBTEXT	COBUSL	COBLIST	COBMAST	COBNEW
RPG	RPGTEXT	RPGUSL	RPGLIST	RPGMAST	RPGNEW
BASIC	BSCTEXT	BSCUSL	BSCLIST	N/A	N/A

Table-2

PARM=Value	Textfile	Listfile	Uslfile	Masterfile	Newfile
0	-	-	-	-	-
1	X				
2		X			
3	X	X			
4			X		
5	X		X		
6		X	X		
7	X	X	X		
8				X	
9	X			X	
10		X		X	
11	X	X		X	
12			X	X	
13	X		X	X	
14		X	X	X	
15	X	X	X	X	
16					X
17	X				X
18		X			X
19	X	X			X
20			X		X
21	X		X		X
22		X	X		X
23	X	X	X		X
24				X	X
25	X			X	X
26		X		X	X
27	X	X		X	X
28			X	X	X
29	X		X	X	X
30		X	X	X	X
31	X	X	X	X	X

Corrections to Issue 21 of COMMUNICATOR 3000

1. page 37 The version level of IMAGE/3000 was incorrectly stated to be B.02.02. The correct version level is B.02.01.

2. page 39 Module 77-- formerly STKDUMP-- was incorrectly stated to be part of the MPE III Series II/III Software Update, date code 1918. As of the B.01.00 (1918) version of MPE, Module 77 no longer exists. STKDUMP is now combined with ABORTRAP into Module 58, ABORTDUMP.

3. page 99 The version level of IMAGE/3000 was incorrectly stated to be B.02.02. The correct version level is B.02.01.

4. page 107 The version levels of Basic Compiler, RPG, KSAM and IMAGE were incorrectly stated. The correct version levels are:

BASICOMP/3000	B.00.11
RPG	B.04.04
KSAM/3000	A.03.00
IMAGE/3000	B.02.01

We apologize for any inconvenience these mistakes may have caused.

Although every effort is made to insure the accuracy of the data presented in the **Communicator**, Hewlett-Packard cannot assume liability for the information contained herein.

Prices quoted apply only in U.S.A. If outside the U.S., contact your local sales and service office for prices in your country.

Printed in U.S.A. 12/79
Part No. 5951-6113

HEWLETT PACKARD