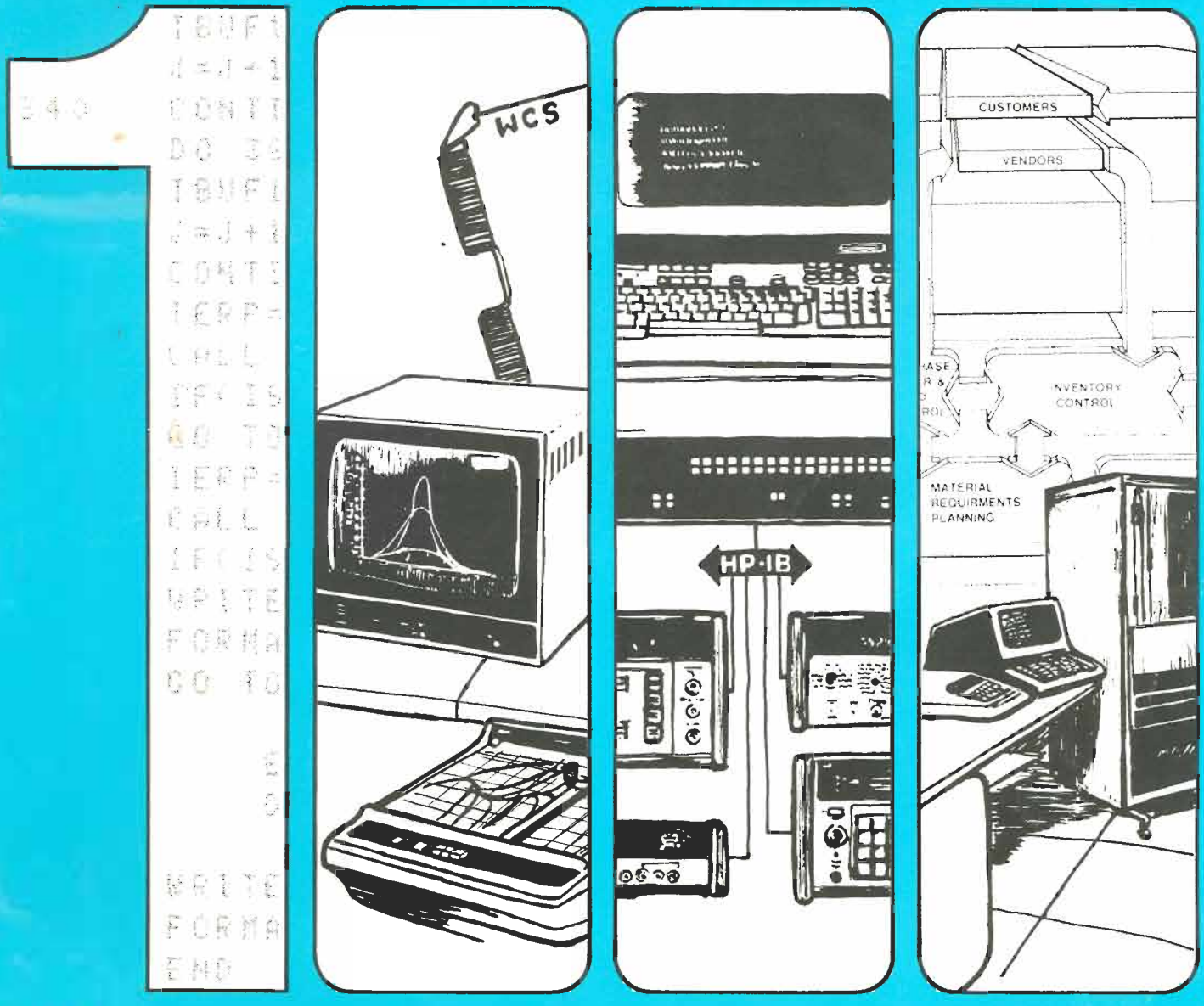
HEWLETT hp PACKARD

Computer Systems

COMMUNICATOR



1978

Volume II Issue 4

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

ANNOUNCING THIS MONTH'S CALCULATOR WINNER

Harvey Bernard, of Hewlett-Packard in Rockville, Maryland, is this month's winner of an HP-21 calculator. Harvey's winning article is Type 6 Files and appears in this month's issue of the Communicator.

Harvey's article was judged to be the best based on the areas of clarity, completeness of subject coverage and interest to largest segment of our readership. These three points are the criteria on which we hope to judge future articles, also.

Alas, there is no calculator winner among our customer readership, since there were no entries. We hope to see this rectified in our next issue. We would also like to encourage more articles from the Hewlett-Packard field employees, so next month, there will be a possibility of three winners: customer, HP Data Systems Division (Division 22) and HP Other.

The eligibility rules are:

1. The calculator is awarded to the best feature-length article which falls in any one of the following categories:

   Operating Systems
   Instrumentation
   Operations Management
   Computation

2. Feature-length is defined as at least 1600 words, exclusive of listings and illustrations.

3. No individual will be awarded more than one calculator per calendar year.

4. In the case of multiple authors, the calculator will be awarded to the first listed author of the winning article.

5. An article which is part of a series will compete on its own merits with other articles in this issue. The total of all articles in the series will not compete against the total of all articles in another series.

6. Employees of Technical Marketing in the HP Data Systems Division (Division 22) are not eligible.

All entries will be judged by a team of at least three people in Technical Marketing.

The deadlines for articles for the next two issues are:

Volume 2, Issue 6 — October 15th
Volume 3, Issue 1 — December 15th

All winners are announced in the HP 1000 Communicator in which their winning article appears.

This is also a good time to talk about our recent addition — OEM Corner. This section is for HP customers who market software of their own development for use on HP 1000 systems. The software may be part of a system package which the OEM delivers as a "turnkey" package or a stand-alone software package. HP has many quality OEMs whose products often address markets which are specialized or aimed at a specific application area. Therefore, these products complement the systems offered by HP itself.

We have no article for our OEM corner in this issue. However, we are looking forward to receiving some contributions from many of our fine OEMs.

To qualify for inclusion in OEM corner, an article should be of general interest to our readers and have educational value. By that we mean it should describe a technique or method of doing something. The article should contain examples and be application-oriented rather than theoretical. We encourage the OEM to describe the features of his product, but remember that the purpose is to educate our readers.

EDITOR'S DESK

The OEM is encouraged to place at the end of the article up to 150 words of purely commercial information. This may include prices of the product and ordering information.

The article should be a minimum of four typed, double-spaced pages. We would prefer that the maximum length not exceed ten pages, but this restriction can be waived.

Deadlines for specific issues of the Communicator are the same as those mentioned above for the calculator entries. Address all communications to

Editor HP 1000 COMMUNICATOR
Hewlett Packard Data Systems Division
11000 Wolfe Road
Cupertino, California 95014
Building 42U

All communications should include the author's address and phone number.

If possible, include the text of the article in machine-readable form, i.e., a file on magnetic tape, mini-cartridge or paper tape.

CONTENTS

The User's Queue includes announcements of new programs added to the Contributed Library (LOCUS) and tips, techniques or methods suggested by our readers.

NEW CONTRIBUTED PROGRAMS

The information below serves as an update to the Data Systems LOCUS Program Catalog (22000-90099).

The new contributed programs listed below are now available. Contact your local HP Sales Office to order Contributed Library material, or (if you are in the U.S.), you can use the direct mail order form at the back of the Communicator 1000.

Please note the following corrections to previous entries in the catalog. LOCUS programs 22658A-K21 and 22682-10993 are not available on mini-cartridge medium.

22682-18996 E-SERIES MICROCODED FAST FOURIER TRANSFORM

This subroutine performs a microcode enhanced FFT (Fast Fourier Transform) with automatic scaling on overflow.

Data to be transformed must be presented as a single integer array. The data may be complex or real. If complex, the odd subscripted elements are the "reals", and the even subscripted elements are the "imaginaries". If real, the "reals" are contiguous.

A complex transform of size N requires 2*N memory locations. A real transform of size N requires N memory locations.

Execution time for 1024 complex points is about 200 ms. Execution time for 1024 complex (all imaginaries = 0) points or 1024 real points is about 100 ms. Each overflow increases execution time by 4 ms for complex transforms; each overflow increases execution time by 2 ms for complex (all imaginaries = 0) or real transforms.

The assembly language portion of the subroutine requires 269 memory locations. The microcoded portion requires 512 words of control store.

There are three calling sequences: complex, complex (all imaginaries = 0), and real.

22682-18996 PT $40

22682-13397 LIST HP-IB CONFIGURATION (CNFG)

CNFG is a program which points out the current HP-IB configuration in an HP1000 system. The following calling sequences are available:

RU,CNFG — for printing all HP-IB EQT's and LU's
RU,CNFG,input,list,EQT — for a particular EQT and all associated LU's
RU,CNFG,,,,LU — for a particular LU
RU,CNFG,,,EQT,LU — for a matching EQT and LU

CNFG must be compiled on the 1805 revision of FTN4 or later.

22682-13397	mini-cartridge	$40

22682-13399 OBJECT MODULE LISTER (LUCY)

This program inputs a type 5 binary relocatable file name (namr) and produces a consolidated listing of its characteristics. The listing includes program name, type, extended NAM record, local length, common length, base page length, producer, all entry points, all externals, EMA record, and transfer address if a main program. The program can be run in batch or conversational mode and has extensive error checking, including BREAK capability.The listing format is suitable for easy-to-read documentation.

22682-13399	mini-cartridge	$35

22682-13398 RTE PROGRAM TIME SHARE PROCESSOR (TMSHR)

This program allows the RTE user to time-share up to 64 compute-bound programs, i.e., it serves as an on-line Real-Time Execution Monitor. The user runs the program with the RU or ON command to pass it configuration information or place it in the time-list for periodic execution to perform the time-sharing function. While in time-share mode of operation, the program time-slices program execution of all entered program names according to each program's configuration information. The program takes only 2K words of memory and can run in any area of RTE, although memory-resident is suggested.

22682-13398	mini-cartridge	$35

LETTERS

We have lots of letters from our readers this month. It's good to see. The first letter is from ***John Durgin,*** who has both correction to "A Solution to the Multi-Terminal Blues" and more information for us.

"The article entitled "A Solution to the Multi-Terminal Blues" (Issue 16) was a great help for our lab, where we really have multi-terminal blues! However, in implementing the transfer file (page 11) suggested, we discovered one further 'bug', in addition to the correction of Volume II, Issue 2. When the transfer file terminates due to lack of blank ID segments, the type 6 file is not renamed to EDI00. When the transfer file is run again, it hangs up in a loop between statements 8 and 11, since EDI00 cannot be found. The solution is simply to add the following directive after the AN message of statement 21.

```
:RN,3G:SC:-2,EDI00
```

Also, the last line should be changed to:

```
:IF,,EQ,,-7
```

Being short of long ID segments, we opted to utilize this transfer file for several programs and have found it most helpful! Thanks!

Here's another suggestion for people with multi-terminal blues. When FORTRAN programs encounter errors in execution, e.g., library errors, formatter errors, pause and stop messages, those messages are printed on certain default devices. This problem can be solved with the following assembly language subroutine and suggested FORTRAN calling sequence.

Keep up the good work in the COMMUNICATOR!

Sincerely,
John Durgin
Fluid Dynamics and Diffusion Lab
Colorado State University
Fort Collins, Colorado 80523"

```
ASMB,R,L
NAM    LIBER,7
       EXT ER0.E,.ENTR,PAU.E,FMT.E
       ENT LIBER
LU     BSS 1
LIBER  NOP
       JSB .ENTR
       DEF LU
       LDA LU,I
       STA ER0.E
       STA PAU.E
       STA FMT.E
       JMP LIBER,I
       END

FTN4,L
       PROGRAM PROGA
       :
       :
       DIMENSION IPAR(5)
       CALL RMPAR(IPAR)
       LU=IPAR(1)
       IF(LU.LT.1)LU=1
       CALL LIBER(LU)
       :
       :
```

Many thanks for the correction, ***John,*** and for the compliment, too.

The next letter, from *C. C. Skelton,* also contains a correction to a previous article.

"In a recent issue of the COMMUNICATOR (Issue 14), in the 'Bit Bucket' section, ***Jim Bridges*** described an algorithm to give programs the capability of using file manager calls on an LU. A problem occurs if the LU passed to the algorithm is associated with a subchannel other than zero. The equation:

```
IEQT=IGET(IDRT+LU-1),
```

which is supposed to return the equipment table address of the channel, also has in bits 11 through 15 the subchannel number. This leads to an invalid offset into the equipment table. Masking out bits 7 through 15 corrects the problem.

Sincerely yours,
C.C. Skelton
Eli Lilly and Company
Clinton Laboratories
Box 99
Clinton, Indiana 47842"

Again, thanks for the correction. Our readers are really sharp.

John Connor has a suggestion for all of us to try.

"I have been using a technique for passing mixed argument types to general purpose subroutines, which I believe to be both efficient and easy to code.

Given an application, say a numeric to alpha conversion subroutine, the normal argument list might be:

```
SUBROUTINE NTOA(IVALUE,RVALUE,DPVALU,ITYPE,STRING,LENGTH)
```

The above arguments being defined as IVALUE=integer value to be converted, RVALUE=real value to be converted and DPVALU=double precision value to be converted. ITYPE would define which one to use.

```
SUBROUTINE NTOA(DPVALU,ITYPE,STRING,LENGTH)
```

In this case DPVALU is type as double precision and can be processed as follows:

```
      SUBROUTINE NTOA(DPVALU,ITYPE,STRING,LENGTH)
      INTEGER STRING(LENGTH)
      DOUBLE PRECISION DPVALU,DVALUE
      EQUIVALENCE(DVALUE,VALUE,IVALUE)
      DVALUE=DPVALU
      N=ITYPE
      GO TO (10,20,30),N
C     PROCESS A DP-VALUE
C     USING DVALUE
10    CONTINUE
      :
      :
C     PROCESS A REAL VALUE
C     USING VALUE
20    CONTINUE
      :
      :
C     PROCESS AN INTEGER VALUE
C     USING IVALUE
30    CONTINUE
      :
      :
```

In the above code, three words are moved to DVALUE starting with the word address passed. Given the proper code in ITYPE, the user need not care what is moved beyond what is needed. (Providing something is there.)

It is equally easy to return values this way and only a minor change is necessary. Assume in this case we are going to extract a value from a string. The routine might look like this.

```
      SUBROUTINE ATON(STRING,LENGTH,ITYPE,IVALUE)
       INTEGER STRING(LENGTH),IVALUE(3),IV(3)
       DOUBLE PRECISION DPVALU
       EQUIVALENCE(DPVALU,VALUE,IV(1))
C      FIRST SECTION WOULD
C      DETERMINE DPVALU,VALUE OR
C      IV(1) BASED ON ITYPE
       :
       :
C      BRANCH TO PROPER
C      RETURN
       GO TO (100,200,300),ITYPE
C      RETURN A DP-VALUE
100    DO 101 I=1,3
101    IVALUE(I)=IV(I)
       RETURN
C      RETURN A REAL VALUE
200    DO 201 I=1,2
201    IVALUE(I)=IV(I)
       RETURN
C      RETURN AN INTEGER VALUE
300    IVALUE=IV
       RETURN
       END
```

I should point out that the user should be careful using the first example. Depending on where the first word is stored, it is possible to create a memory protect violation. Should the last word of a labeled common block or system common block be passed, an error could possibly occur. It is possible to avoid the possibility of this by setting up an equivalence in the calling routine and moving the value there, always passing the local variable as the argument.

I hope this will be of use to COMMUNICATOR readers as I have found many features useful to me.

Yours truly,
John Conner
Digicon Geophysical Corporation
3701 Kirby Drive
Houston, Texas 77098"

I hope so, too, ***John.*** I'm sure it will be. Many thanks to all our correspondents for their suggestions.

RTE-IV — GETTING ONE'S ACT TOGETHER

David L. Snow and Shaila P. Kapoor

This is the second in a series of articles by the RTE-IV development team describing the inner workings of the RTE-IV operating system. These articles go into some depth and therefore assume that the reader has studied and become familiar with the material in the RTE-IV reference manuals. The article on RTE-IV memory structure in Volume II, Issue 2 of the COMMUNICATOR/1000 will be of use in reading this article.

The RTE-IV operating system doesn't, as many people assume, just spring into life at bootup time. Besides going through a several stage bootstrapping process, it must, among other things, initialize various lists, establish constants and maps, begin the time-base generator and, optionally, allow the user to modify his I/O and/or memory configuration. These processes can be divided into three areas which we will discuss in some detail:

RTE-IV Bootup
RTE-IV I/O and Memory Reconfiguration
RTE-IV Operating System Startup

RTE-IV BOOTUP

"Bootstrapping" is a process by which the user reads into memory a short series of instructions which when executed, will read into memory a larger set which may be used to read in an even larger set. The HP1000 computers (formerly known as the 21MX computers) include a series of ROM boots which, when loaded into memory by the computer front panel, will execute the code loaded into the last 64 words of the first 32 pages of memory. For disc ROM boots (also known as Basic Binary Disc Loaders) these 64 words of code will load into memory beginning at location 2011_8 the 128 words stored at track 0, sector 0 of the specified disc surface. These 128 words are known as the boot extension. The disc boot then transfers control to location 2055_8 or relative location 44_8 in the boot extension.

For RTE-IV systems, the boot extension will be used to read into the first 32 pages of memory (see figure 1) from track 0, sector 2 of the system disc the code representing the system base page, table area I, driver partition 1, common, system driver area, table area II, the operating system and $CNFG, the first half of the reconfiguration program which occupies the first chunk of system available memory (SAM). Before doing this however, the boot extension must accomplish two tasks. First it checks the front panel switch registers to determine if bit 5 was set at bootup. If set, the user is indicating that he wishes to do I/O or memory reconfiguration so the boot extension executes a HLT 77. At this point the user may indicate a modification of the system disc and/or system console I/O configuration by setting bit 15 of the switch register and inserting the disc's select code into bits 6-11 and the system console's select code into bits 0-5.

Once RUN is pushed, the boot extension accomplishes its second task — moving itself to location 77500_8 of the thirty-second page of physical memory. This is required since the boot extension will be loading the system into the lower 31 pages. After moving itself out of the way, the boot extension configures its disc I/O instructions to any new disc select code and then loads into memory the operating system from the base page up through the configurator program (first part of SAM). If the load is successful, the boot extension will execute a JMP 3,I. Since the generator has stored the starting location of the operating system ($STRT in the Scheduler) into location three, this will cause an entry into the system's startup code.

If the user did not specify I/O or memory reconfiguration by initially setting bit 5 of the switch register, then the process is the same except that the HLT 77 is not executed and the generator specified select code for the system disc is used instead.

RTE-IV I/O AND MEMORY RECONFIGURATION

Before passing control to the I/O configuration program, $STRT intializes the following memory bounds constants contained in table area II:

$CMST — Logical/physical starting page of common.
$COML — Number of pages of common.
$SDA — Logical/physical starting page of the system driver area.
$SDT2 — Number of pages occupied by the system driver area and table area II.
$RLB — Logical starting page of the memory resident library.
$RLN — Number of pages of memory resident library.

After setting these constants, $STRT sets up the system map to reflect the prereconfiguration memory layout, enables the map with a SJP instruction and transfers control to the configuration program $CNFG, which was loaded by the boot extension into the first chunk of SAM.

The configurator program is divided into two parts which are relocated as separate programs, $CNFG and $CNFX, by the generator. The first part of the configurator, $CNFG, is relocated as a type 16 program. Type 16 is a special type used only for the configurator which signals the generator to load $CNFG as a system module overlaying the default SAM. The second part of the configurator, $CNFX, is relocated by the generator as a type 3 background disc resident program. It is loaded into memory by $CNFG under a background disc resident program map (see Figure 2).

The configurator program is divided to make more memory available to it since it cannot entirely fit in the first part of SAM. By making $CNFX a type 3 program, it can access all the system entry points and thus communicate with $CNFG. $CNFG has to reside in the first part of SAM as a system module, so that it can use the $XSIO routine in RTIOC for input and output. EXEC calls cannot be used since the system has not been initialized. The work load divided between $CNFG and $CNFX is based only upon how much code can fit in the first block of SAM. $CNFG will load the memory resident and driver partitions, reconfigure I/O and contain the I/O subroutines also to be used by $CNFX. $CNFX will handle memory reconfiguration.

The $CNFG program clears all interrupts as soon as it is given control. It then saves the base page locations SYSTY (EQT entry address of the system TTY), DUMMY (privileged I/O card location) and EQT1-EQT10 and clears them. Clearing SYSTY prevents the user from gaining control of the system by striking a key on the keyboard of the system console and getting a prompt. DUMMY is cleared to prevent any privileged interrupts. SKEDD is cleared and $LIST is set to 1, to prevent any scheduled programs from running. These locations will be restored just before $CNFG returns control to $STRT. If the console or list device is buffered, $CNFG will clear the B bit in word 4 of the device's EQT to make it unbuffered so that SAM is not needed for I/O. The original buffered or unbuffered status will be restored before returning control to the system. I/O errors cannot be handled due to the fact that operator console capability is taken away from the user. If an I/O error does occur, the boot-up procedure will have to restart. The $CNFG program will pick up the new select code (if any) for the system disc from the switch register and reconfigure it in memory so that disc accesses can be made for loading the memory resident area and driver partitions. Since the disc driver is required to load these additional areas, the generator forced the system disc driver into driver partition 1. This driver partition is the only one loaded into memory by the boot extension (see Figure 1).

I/O reconfiguration is performed in $CNFG by assigning the current select code's trap cell and interrupt table entry to the new select code obtained by prompting the user for changes. The equipment table entry pointing to the current select code is changed to point to the new select code. Initially, the changes needed to be made for I/O reconfiguration are recorded in tables in $CNFG's area of memory. At the end of the I/O reconfiguration, these changes are transferred to the trap cell, interrupt table and equipment tables in the system in memory. To enable the configurator to load the driver partitions and memory resident programs and also to perform I/O and memory reconfiguration interactively, the system disc, system console and the list device select code configurations have to be changed in the actual tables in the system in memory before the reconfiguation process can begin.

To optionally make I/O reconfiguration permanent, the following tables and base page locations must be written out on disc: interrupt table, trap cells, fourth word of all EQT's, first word of the device reference table (DRT) (which is the entry for the system console) and base page locations TBG, SYSTY and DUMMY.

Prior to writing the interrupt table on the disc, its entries for the new console and list device select codes are saved. These are replaced with what the entries were before any I/O was performed to these two devices. The saved entries are restored after the interrupt table is written on disc. This is done because some of the console and list device drivers when executed for the first time, change the interrupt table entries.

$CNFG now enters the memory reconfiguration phase. The user is allowed to define up to 100 bad pages of memory in the user's area. The number of pages in the SAM extension as it is currently defined are determined from the system entry point $MPS2. The physical starting page number for SAM extension is determined by adding contents of $ENDS, number of pages taken up by the driver partitions, number of pages for memory resident base page, library and programs. $CNFG then checks to see if this resulting starting page of SAM extension is included in the list of bad pages. If so, the start page of the SAM extension is incremented to avoid the bad page (and any other consecutive bad pages). If $MPS2 is not zero, this start page is compared with it. If they do not match, $MPS2 is changed to match this newly evaluated starting page. This case may happen if some previously bad pages at the start of the SAM extension were replaced with a new memory module or some pages at the start of SAM extension went bad.

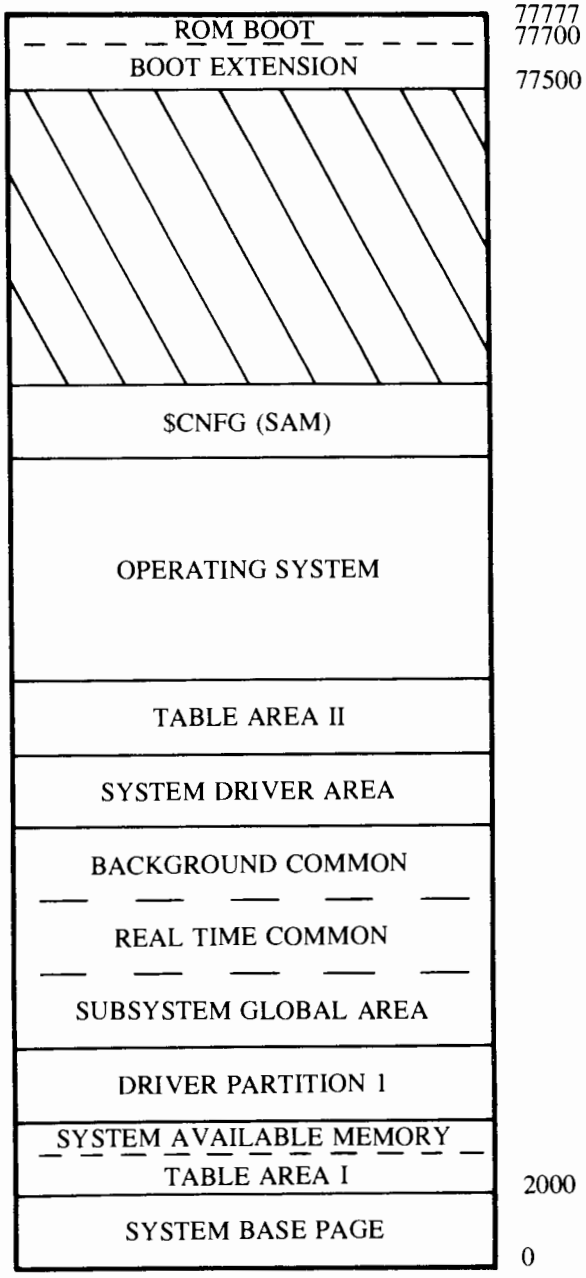


Figure 1. Addressing Space for $CNFG

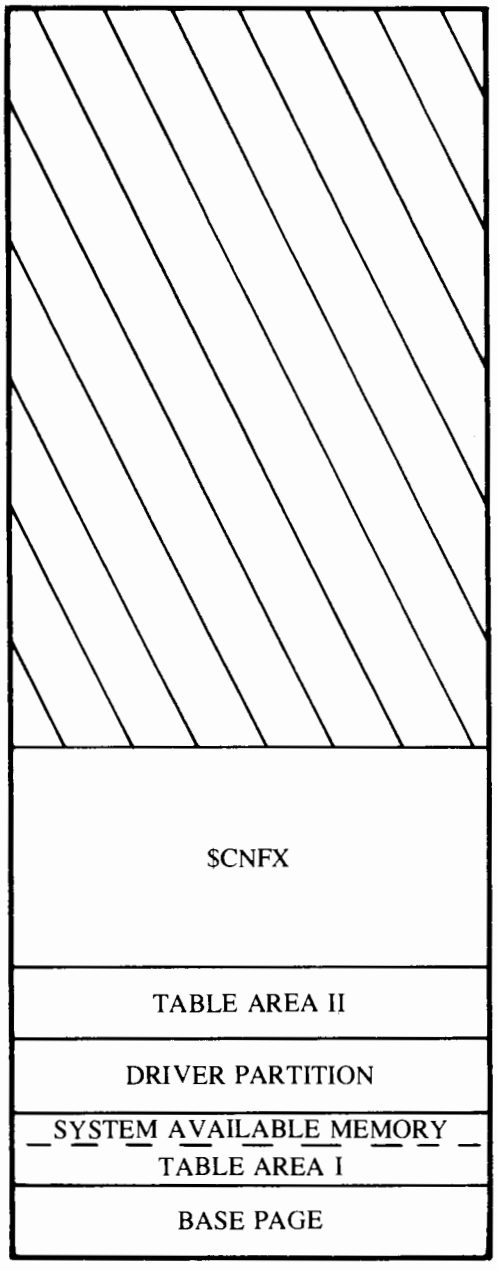


Figure 2. Addressing Space for $CNFX

The user is now given the chance to increase the size of his SAM extension. $CNFG will allow the user to define a SAM extension which is broken up into as many as five contiguous areas by bad pages. The system map is modified to reflect any changes caused by modifications to the SAM extension.

$CNFG now transfers control to the configurator extension, $CNFX, which prompts the user for redefinition of partitions, the assigning of programs to partitions and the reserving of partitions. $CNFX picks up blocks of memory between bad pages or pages remaining if there are no more bad pages and prompts the user to define partitions for it. Partitions defined for a particular block of memory must use up all the pages in the block, otherwise $CNFX asks the user to redefine partitions for that particular block. $CNFX fills the start page of the partition, number of pages, reserved bit R and RT or BG values into the memory allocation table (MAT) entry for the partition. If the number of pages in the partition is larger than the maximum addressable partition size, $CNFX asks if subpartition definition is desired. If the response is yes, the partition just defined becomes a mother partition and the M bit in the MAT entry is set. Every partition following with an 'S' as the fourth parameter is a subpartition. The block of memory to be allocated to subpartitions is the same as that used by the mother partition. The subpartition link word (SLW) of the mother partition is made to point to the first subpartition, SLW of the first subpartition points to the second and so on. SLW of the last subpartition points back to the mother partition.

After all the partitions have been defined, $CNFX threads them into either real time, background or chain partition (mother) free lists. Several passes have to be made through the MAT table to thread each list. $MCHN, $MBGP, $MRTP, $BGFR, $RTFR, and $CFR entry points are set up by $CNFX while threading these lists.

To optionally make memory reconfiguration permanent, the memory allocation table, ID segments, and system entry points $MCHN, $MBGP, $MRTP, $BGFR, $RTFR and $CFR are rewritten on the disc. Finally, the configurator program changes the name of $CNFX to ",,,,," so that the configuration extension cannot be run online and returns to the system startup routine.

RTE-IV OPERATING SYSTEM STARTUP

Once control is returned to $STRT from the configurator programs, the system resumes its initialization. Code for the startup process is contained in almost every module of the operating system. Since this code is executed once, buffers which will be destroyed during system operation are used to execute most of the code. RTE-IV first calls $RTN to collect all available SAM and return this memory to the system. The generator stores information on up to five chunks of SAM in base page words EQT1 to EQT10. Each two words represent one area of SAM with word 1 indicating the logical address and word 2 (which may be zero) indicating the length. Currently the generator includes pointers to three chunks of SAM — the main area of SAM where $CNFG was loaded, the SAM extension, and the unused portion of the last page of table area I. $RTN will link the chunks of SAM into a list in order of increasing size and increment a counter indicating the "maximum SAM now". When all of SAM is collected, RTE-IV transfers control to $ALC where the "maximum SAM now" is posted as the "maximum SAM ever".

After setting up SAM, RTE-IV initializes the non-system map registers. The user map is set to reflect the memory resident program map, while the Port A and B maps are set up to reflect the system map. Pointers are set up to allow the future swapping of extended memory areas in chunks equal to the area between the start of common and the end of the user map. The base page fence is set up as the last user link plus one with the lower portion mapped.

Next, RTE-IV initializes the pointers to the memory allocation table which defines partitions. If no real-time partitions exist, the background list of partitions will be used for real-time programs. If no background partitions exist, the real-time list of partitions will be used for background programs. If no mother partitions exist, the background list of partitions will be used for programs requiring mother partitions. These algorithms assume that either the background or real-time partition lists is not empty.

The swap delay is now obtained from the base page word SWAP where the generator placed it, negated and stored for use by the Dispatcher. Next, the minimum track size in sectors between the two system discs, LU 2 and 3, is computed and saved to be used in swapping programs. The program FMGR is now scheduled and forced to the head of the scheduled list in front of any startup program define by the generator. FMGR's priority is set to 0 with its true priority stored in its ID temporary word 2. The length of the track assignment table, which is stored in base page word TATLG, is set to −1 with its true value stored in FMGR's ID temporary word 1. By setting the track assignment tables's length to −1, all track allocations will be inhibited until the FMGR can properly set up the file system. The FMGR will restore the track assignment table's length and FMGR's priority when it initially executes.

The ID addresses for D.RTR, FMGR and SMP are now computed and saved for future use by the system. The list of user partitions are checked to see if there is a partition of sufficient size to run FMGR or D.RTR. If not, a HLT 10 is executed. This situation can exist since the reconfiguration process allows the user to redefine his partitions at bootup. The constant D$RN, giving the address of the resource number table, is made direct. The time base generator is started and the "SET TIME" message is sent to the system console. The privileged I/O card is initialized, the disc protect option ($PDSK,AB,1 at generation) is set up and a HLT 2 and HLT 3 are stored in locations 2 and 3 of the system map. The system now transfers control to $XEQ in the dispatcher to begin normal system operation. At $XEQ the following three steps will occur:

1. Process interrupts for the "SET TIME" message. The user should set the real time clock with the TM command or it will default to 8:00 A.M., January 5, I978.
2. Dispatch FMGR which will setup the file system, restore his priority and base page word TATLG, and try to transfer to the file WELCOM.
3. Execute any user defined startup program.

Having done all the items discussed in this article, the system considers that it has its act together and allows the user to begin executing his programs. In the next issue, we will discuss Extended Memory Arrays or "How to fit a million words into two thousand locations".

TYPE 6 FILES

Trying to explain the difference between files and programs to a beginner is worse than Chinese water torture. The poor novice has apoplexy when you reveal that files can contain a program's source as well as object code. So you give him heart massage and this bit of advice: programs are executable, files are not. He is fine until he discovers that type 6 files are executable! Then the neophyte experiences acute coronary pain. I suggest you give him a copy of this article.

WHAT DO THEY LOOK LIKE?

A type 6 file contains memory image (executable) code, and its sole purpose in life is to be "run." Like any file, a type 6 lives in a FMGR area (a cartridge) on the disc. Like other files, it has a directory entry (see figure 1) pointing to it. The first block (128 words) of the type 6 contains information normally found in a program's ID segment: program priority, type, etc. The remainder of the file is an exact copy of a program which has previously been loaded.

When it is created, a type 6 looks like a type 3, 4, or 5 file: an EOF mark, a −1, is neatly tucked into its first word. However, in order not to corrupt its memory image code with the length words that normally bracket variable length records, type 6's are accessed as and look like a type 1. Thus, a type 6 is a hybrid among files, as you can see in figure 2.

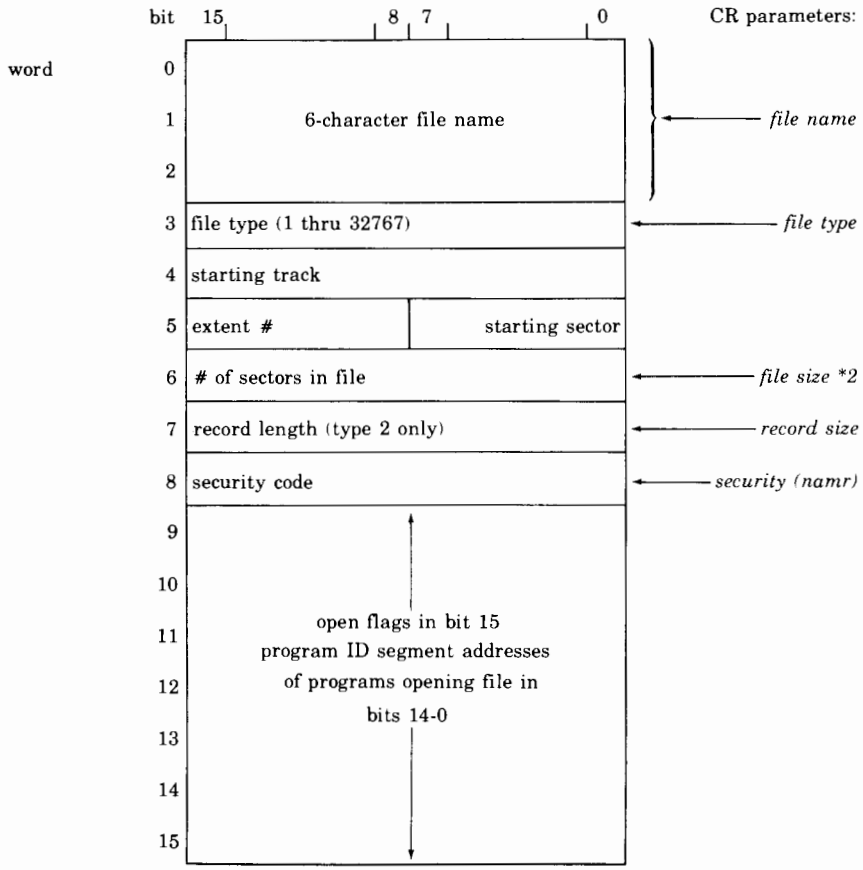


Figure 1. Disc File Directory Entry

Memory-Image Program File Formats (Type 6)

Files created by the SP command as memory-image program files are always accessed as type 1 files (fixed length, 128-words per record).

	Word	Content	
1st two sectors contain program's ID-segment information	0	-1	← EOF unless forced to type 1
	1-5	not used	
	6	priority	
	7	primary entry point	
	8-13	not used	
	14	program type	
	15-16	not used	
	17-19	time parameters	
	20	substatus 1 - word 21 of ID segment	
	21	substatus 2 - word 22 of ID segment	
	22	low main address	
	23	high main address	
	24	low base-page address	
	25	high base-page address	
	26-27	not used	
	28	checksum of words 0 - 27	
	29	setup code word	← sum of contents of words 1650 thru 1657 and words 1742 thru 1764 in base page
	30-127	not used	

Remainder of file is an exact copy of the program being saved.

Figure 2. Memory Image Program File Formats (Type 6)

PROS AND CONS

Users often store programs like EDITR and FMGR into type 6 files, Doing so enables the user to create multiple copies such as FMG01 and EDI01, without having to create multiple load modules. We will expand on this subject later.

Since files do not disappear on boot-up, type 6's are useful for making load modules permanent. Furthermore, if you are short of track pool area on your disc, type 6's effectively move your load modules out of the track pool into the FMGR area on LU's 2 and 3. Of course, this is a trade-off; you may, in fact, be tight on FMGR area.

Another advantage of a type 6 over a load module in the track pool, is that it saves space. The on-line loader allocates space in tracks, whereas the FMGR allocates file size in blocks. Consequently, a load module of, say, 127 words would waste 6017 words (one track minus 127 words) in the track pool, but only use one word as a type 6 file! (This excludes the 128 word ID block and the 16 word directory entry.)

On the other hand, a type 6 involves a minor inconvenience. It must be executed under FMGR or must be RP'ed, a command we will discuss in the next section.

HOW TO USE THEM

Let's begin by loading a program with the on-line loader. Figure 3 shows you both a temporary and permanent load. The temporary creates memory image code on disc, pointed to by an ID segment in memory. The permanent does the same but also creates an ID segment on disc. For this example, assume we have a temporary load. Under a FMGR copy, do a :SP,PROGX (Save Program command), where "PROGX" is the name of the program you loaded. The :SP causes a search to be made of the ID segment table for "PROGX", and results in the code and selected ID information for "PROGX" being stored ina type 6 file name "PROGX." Now :OF,PROGX to eliminate the loaded version and you have the situation represented in figure 4. (To eliminate a permanent program, you would RU,LOADR,,,4.) You should note that by default your file will be on LU's 2 or 3 and that only type 6 files on LU's 2 or 3 can be executed. A :SP,PROGX::-12, for instance, would be good for nothing but temporary storage. Under your FMGR copy you can now :RU,PROGX. Under FMGR, the "RUN" command works as follows: first, it initiates a search for an ID segment named PROGX; second, if no ID segment is found, FMGR looks for a type 6 file named "PROGX"; third, FMGR will look for any file named "PROGX" and transfer to it. Hence, if there is no such program or file, you end up with a FMGR -006 (file not found). In our example, having found a type 6, FMGR will locate an empty ID segment, copy data into it from the first block of the file, and schedule the program. When PROGX terminates, the ID segment data will be erased, making it available again. Because there is no longer an ID segment, you could not exit FMGR and RU,PROGX.

If you want to run PROGX outside FMGR, you must :RP,PROGX (Restore Program) which creates an ID segment in memory pointing to the file. On boot-up, you must :RP,PROGX again to reinstate the ID segment. When you start running out of ID segments, you can :RP,PROGX,PROGY, which will erase PROGY's ID and replace it by PROGX's. This assumes PROGY was previously RP'ed. Figure 5 illustrates the above commands.

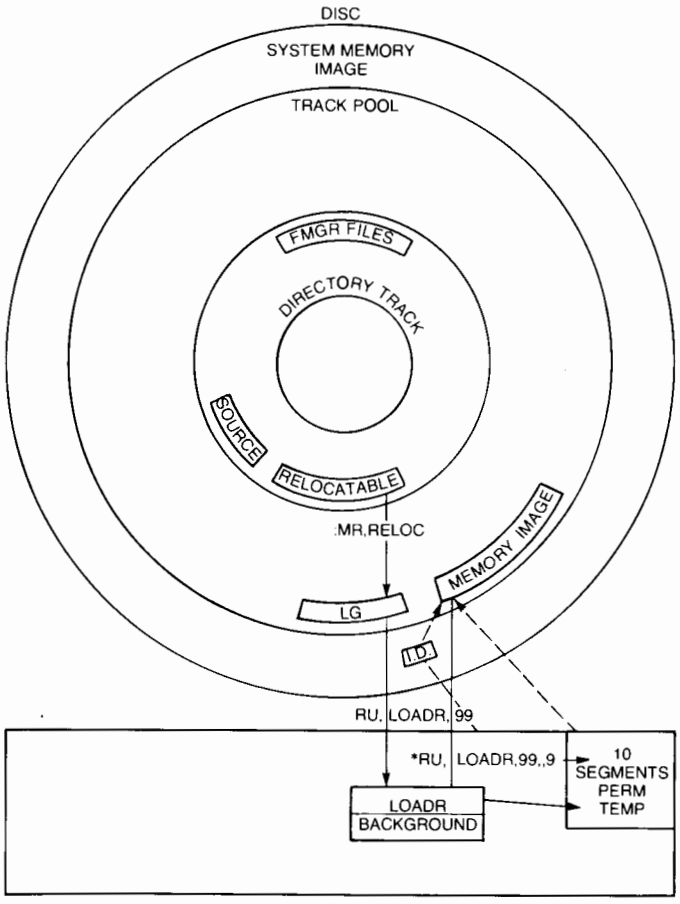


Figure 3. Loading Program

CREATING COPIES OF PROGRAMS

If we want to create two copies of our load module, to be called PROG1 and PROG2, we do the following -- :RN,PROGX,PROG1 so that the name of our type 6 file is "PROG1," then :RP,PROG1. Now :RN,PROG1,PROG2 and :RP,PROG2, so that we have two ID segments in memory, one for PROG1 and one for PROG2, both pointing to the same memory image code.

Under RTE-II/III, this technique is used to create copies of FMGR and EDITR. To insure that these copies exist after each boot-up, the :RN's and :RP's would be done in the WELCOM file.

TYPE SIXING A SEGMENT

In the above discussion nothing was said about copying FMGR's segments: FMGR0, FMGR1, etc. Since each FMGR copy loads the same segments, it suffices to have one version of these on disc; in this case the permanent version created at system generation. If you decide, however, that you want to store some temporary segmented program into the FMGR area, you will have to :SP the main and all its segments. How does the main find the segments when you say :RU,MAIN? Unfortunately, it cannot, until you create pointers to the segments by :RP'ing them.

The preceding is a powerful tool when used with transfer files. In a procedure, for example, you can :RP your segments, :RU,Main, and then :OF the segments, thereby releasing all the ID segments you used.

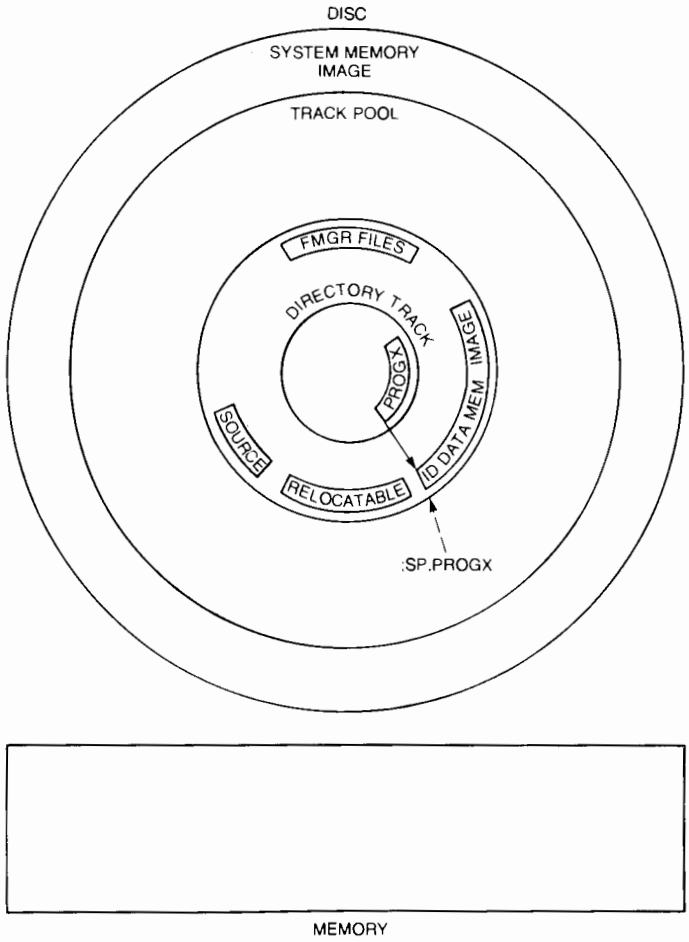


Figure 4. Offing Program

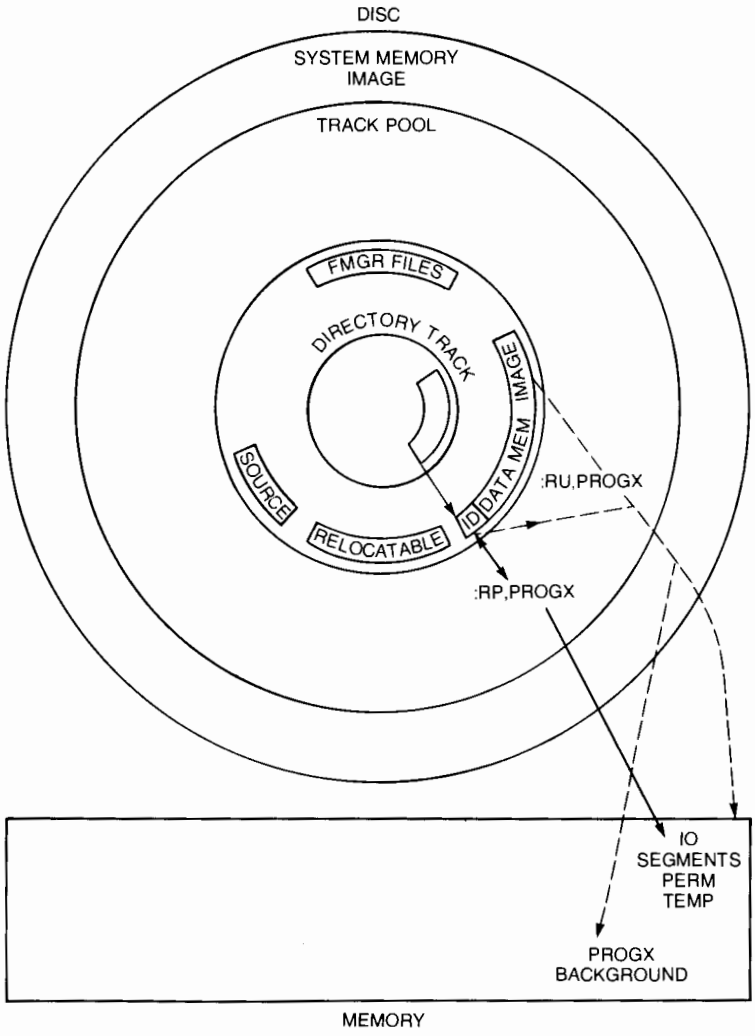


Figure 5. RPing Program

MISCELLANY

Here are several type 6 gotcha's. When you try to :PK,-2 or :PK,-3, you may get a FMGR -011 and a list of programs which have been RP'ed. You have to :OF these programs before the :PK. RTE is afraid the RP'ed ID segments will end up pointing to the wrong files and executing these could wipe out the system.

For the same reason, RTE only lets you run files on LU's 2 or 3, where they are least likely to be overwritten and cause havoc. Furthermore, programs are loaded on one system could spell disaster on another, so RTE will not let you transport a type 6 between systems. Before it can be executed, a code word in the type 6 (see figure 2, word 29) must match a word in the master cartridge directory on LU 2.

Finally, here are a few neat things you can do with type 6's. Using our "PROGX" example, you could :SP,PROGX1, giving it a six character file name, and effectively a six character program name when it is run under FMGR.

You could put two versions of the FTN4 compiler on your system, if you :SP the mains and associated segments with different file names. Consider the two transfer files below:

```
 /FTN4O  (RESTORE OLD FTN4)
:RN,FTN4O,FTN4
:RN,F4.0O,F4.0
:RN,F4.1O,F4.1
:RN,F4.2O,F4.2
:RN,F4.3O,F4.3
:RP,FTN4
:RP,F4.0
:RP,F4.1
:RP,F4.2
:RP,F4.3
:RN,FTN4,FTN4O
:RN,F4.0,F4.0O
:RN,F4.1,F4.1O
:RN,F4.2,F4.2O
:RN,F4.3,F4.3O
:TR
```

```
 /FTN4N  (RESTORE NEW FTN4)
:RN,FTN4N,FTN4
:RN,F4.0N,F4.0
:RN,F4.1N,F4.1
:RN,F4.2N,F4.2
:RN,F4.3N,F4.3
:RP,FTN4
:RP,F4.0
:RP,F4.1
:RP,F4.2
:RP,F4.3
:RP,F4.4
:RN,FTN4,FTN4N
:RN,F4.0,F4.0N
:RN,F4.1,F4.1N
:RN,F4.2,F4.2N
:RN,F4.3,F4.3N
:TR
```

And a similar transfer file would be necessary to release the ID segment and tracks of each of the modules.

How would you like to restrict the usage of certain programs? Just put them into type 6 files with security codes. For example, a :SP,PROGX:-20 would require a :RU,PROGX:-20.

In summary, type 6 files may perplex the beginner, but they offer a novel way to store programs permanently on disc, create copies of programs, rename them, put security codes on them, and much more.

DRIVER WRITING TIPS

John Pezzano, Captain, USAF
Halloman AFB, New Mexico

Here are some tips to use when writing drivers that could save some time in execution.

In order to make a driver reconfigurable and thus able to support more than one device of the same type, HP drivers configure themselves every time they are entered. Is this really necessary? How many users have more than one card reader? If you are writing your own driver, how many thigamajigs are your going to communicate with? Even if the answer is more than one, what are the chances that two of them will be active simultaneously? Probably not very high!

In the RTE Driver and Device Subroutines Manual (92200-93005), the recommended technique is to reconfigure the driver immediately upon entering as follows:

```
I.XX    NOP
        JSB SETIO           Configuration
        .
        .
        .
        JMP I.XX,I          Exit

SETIO   NOP
        .
        .                   Do configuration
        .
        JMP SETIO,I         Exit

C.XX    NOP
        JSB SETIO           Configure
        .
        .
        .
        JMP C.XX,I
```

You can save the twenty to thirty configuration instructions with a little checking beforehand:

```
   I.XX    NOP
(or C.XX)  CPA SC           Same select code as last time?
           RSS              Yes! Already configured.
           JSB SETIO        No! Configure.

   SETIO   NOP              Save select code for next time.
           STA SC
   SC      DEC 0            Initialize at 0 to force first time configuration.
```

You say you are also using DMA? Since there is at least a fifty-fifty chance that you will get the same DMA channel, you can really save a bundle.

```
I.XX    NOP
        JSB SETIO           Do checking in SETIO

SETIO   NOP
CPA SC                      Need to reconfigure I/O channel?
        RSS                 Yes! Skip next instruction
JMP NOSET                   No! Skip configuration
```

Configuration Instructions

```
NOSET   LDA CHAN            Get assigned DMA channel
        CPA DSC             Same as last time?
        JMP SETIO,I         Yes! Exit.
        STA DSC             Save for next time, reconfigure.

SC      DEC 0
DSC     DEC 0
```

Note that the driver is completely reconfigurable but under most circumstances, the configuration is skipped. The added few instructions are more than compensated for by the time saved.

For the real purists, the initial configuration can also be skipped. If SC is initialized to a particular select code and the I/O instructions are also set to that select code, then no configuration is necessary if the preselected select code is the one with which the driver is called. If not, the driver reconfigures. This is particularly true of DMA configuration. Configure all DMA instructions for select code 6 and initialize DSC as 6. If channel six is assigned, no configuration is necessary. If channel seven is assigned, DSC does not compare as equal to CHAN and the driver reconfigures DMA.

If you are writing a privileged driver and want to save time, there are more tricks you can use. The steps for the privileged section of the driver is as follows:

1) **P.XX NOP**
2) Shut off interrupt, save registers, memory protect flag, prevent DMA interrupt,etc.
3) Do I/O
4) Restore everything as before
5) **JMP P.XX,I** to exit

Here are some ways to save time:

1) In an 21MX computer, if you are not using the X or Y registers, don't save or restore them (four instructions saved).
2) If you can get away without using the B register, don't save or restore it (two more saved).
3) If you don't disturb the E or O registers (remember, ADA and ADB instructions set E and O), don't save or restore (eight more!).
4) If you know there won't be any higher priority privileged drivers, leave the interrupt system off the entire time you are in the driver. This saves turning it on and off once (two instructions) and eliminates the need to save and set the memory protect flag. Who cares if it says memory protect is on when you have it off; nothing can interrupt anyway. Remember to check the flag upon leaving the driver and turn on memory protect if it was on when you interrupted. (a few more instructions saved.)

These techniques are simple yet effective. They save time and time is money. The small cost in added instructions for the select code check is more than compensated by the time saved. The elimination of unneeded instructions in a privileged driver can add up to a real difference in system performance.

DESIGNING A DATA BASE FOR MODERN FACTORY MANAGEMENT

by John Koskinen

INTRODUCTION

Let's say that you are a systems analyst in a manufacturing organization and that your production managers are starting to introduce the modern methods of plant management. It becomes obvious that the old manual inventory system or batch computer work order system is completely inadequate to keep up with the demand for proper management information. It looks like the best thing for you to propose is an on-line, data base management system to replace the manual or batch computer systems. The problem that you have is, where to start.

THE PLACE TO START

Contrary to popular belief among some programmers, data base design starts with the people who use the information, not the computer system that processes it. The first design step, then, is to observe and document the way people currently use information in their respective departments. This is not hard to do. In manufacturing companies, virtually all plant activity can be traced by paper forms or computer reports. This leads us to the tangible evidence for starting data base design — the source document. Usually, the information you want to place into the data base is already on some kind of paper form or paper report. The paper form may be a purchase order, inventory KARDEX card, material invoice ticket or QA inspection report. The paper form or report is an excellent place to start because there is already a good deal of thought (intelligent design) placed in the form in terms of data elements, data relationships, and data uses. As a data base designer, you have to translate that paper form into an IMAGE schema. The translation process is not always a simple procedure, especially if the form is complex or is used by a number of different departments. There is, however, a straight forward procedure for defining a data base schema (data base definition language).

Using the paper forms or reports, make a list of all the data items (fields of information) on the form or report. The items always have a title: part number, P.O. number, quantity, date and so on. The data items always have a particular format: alphanumeric, numeric only, dollars and cents, month-day-year, etc. The list of data items should include this format description. Once this list is made, you can ask people in the department what data items are really used, which ones don't need to be in the list, or which additional data items should be included in the data base and hence on the item list.

The next step is to figure out the useful relationships between all the data items. An important observation here is how the people in the department file the paper forms or access the reports. The paper forms may not always be filed in the same sequence during the use cycle of the form. For example, a purchase order may start off in purchase order number sequence as it is being typed up. Then, when the purchase order is sent to the vendor, a copy may be stored in another file by vendor name, and another copy stored in a tickler file by due date.

The order in which the forms are stored tells you how the people in the department like to pull the forms out of the file. Many computer generated reports are reprinted in a different sort sequence. The data item used to sort these forms or reports into a sequence is called a key item. All paper forms have one key item which is unique to that piece of paper: purchase order number, work order number, QA report number, etc. Once you have identified all the key items, you can ask the people in the department what items on the form they use when they pull a copy of the form out of that particular file which has a particular key sequence. For example, an inventory control clerk may want to see all the parts that have been ordered on a purchase order, but doesn't really care about the vendor. On the other hand, a purchasing expeditor, noticing an order is late, first wants to know who the vendor is and then what other orders that same vendor has. In this way, you can fit together a logical set of data items which are most commonly used in various departments. At this point, those sets of items may relate to the original paper form and they may not. What you have done, in data base terminology, is defined a group of items which could be put together to create a data base record.

PURCHASE ORDER

01-E141C 3 PART

4982 REQ. NO. DATE July 10 19 78

TO Hardware Supply

ADDRESS 1024 Second Ave Sunnyvale Ca

SHIP TO Mt. View Storage

ADDRESS 625 Ellis st. Mt. View Ca.

FOR	DATE REQUIRED	HOW SHIP	TERMS
42-63	July 31, 78	UPS	N/A

	QUANTITY	PLEASE SUPPLY ITEMS LISTED BELOW	PRICE	UNIT
1	250	*8 screws, 4203-4123	5 00	Box
2	100	R Angle Brks, 4205-6134	15 00	Gross
3	50	6X4 Plates, 4278-1903	1 00	Ea.
4				
5				
6				
7				
8				
9				
10				
11				

IMPORTANT
OUR ORDER NUMBER MUST APPEAR ON ALL INVOICES-PACKAGES, ETC.

PLEASE NOTIFY US IMMEDIATELY IF YOU ARE UNABLE TO SHIP COMPLETE ORDER BY DATE SPECIFIED.

PLEASE SEND 2 COPIES OF YOUR INVOICE

Smith
PURCHASE AGENT

01-E141C

STORAGE AND RETRIEVAL

The source document — paper form or printed report — has provided design information on data item definition; it can also provide design information on input and output processing. Depending on user needs, data base operation may have to be optimized especially for data input, or data retrieval. The design of the data base should incorporate these performance preferences.

INPUT PROCESSING

Virtually all information in a production facility is found on a piece of paper at some time during its useful life. The odds are very high that the information being placed into a data base as original information being placed into a data base as original information is from a preprinted, manually filled-in paper form. The structure of the form can be used as a guide for designing the various data base records, data sets, and relationships. For example, a purchase order (figure A) has two major types of information: 1) header information which relates mainly to the vendor, order shipping, and terms and conditions; 2) line item information which relates to part-numbers, line item descriptions, prices, and quantities. Obviously, it doesn't make any sense to duplicate the header data for each line item on the order. An optimal design for this purchase order might be one data set for header information and one data set for line item information. The key used to connect these two data sets would be order number. A sample schema is given in figure B. This schema is optimzed for input processing because it only has one key value which is used for both the header and line item data sets. (The more key values associated with a data set, the more system overhead is encountered in storing information into the data base.) An application program serving as the data base input module probably would place a formatted screen on a purchasing clerk's CRT which looked just like the paper purchase order in figure A. The clerk would fill in the blanks, press enter and the program (after doing necessary format, validation and range checks) would create the data base records for that purchase order.

```
          HEWLETT-PACKARD IMAGE/1000 DATA BASE DEFINITION PROCESSOR

$CONTROL LIST,NOROOT,NOSET,TABLE;
BEGIN DATA BASE PUR00;CR015;01;
LEVELS:
      5   CLERK;
      10  SUPER;
      15  MGR;
ITEMS:
    0001      <<    HEADER INFORMATION                    >>
    0001      POHEAD,   U4(5,15);   <<PURCHASE ORDER NUMBER>>
    0002      CDATE,    U6(5,15);   <<PO CREATION DATE     >>
    0003      VENNAM,   U24(5,15);  <<VENDOR NAME          >>
    0004      VENADD,   U24(5,15);  <<VENDOR ADDRESS       >>
    0005      SHPWHS,   U24(5,15);  <<SHIP TO WAREHOUSE    >>
    0006      SHPADD,   U24(5,15);  <<WAREHOUSE ADDRESS    >>
    0007      DEPCOD,   U6(5,15);   <<PURCHASED FOR DEPART >>
    0008      RDATE,    U6(5,15);   <<REQUIRED DATE        >>
    0009      CARRY,    U12(5,15);  <<SHIPPING CARRIER     >>
    0010      TERMS,    U12(5,15);  <<SPECIAL TERMS        >>
    0011      AGENT,    U24(5,15);  <<PURCHASING AGENT     >>
    0012
    0012      <<LINE ITEM INFORMATION                       >>
    0012      POITEM,   U4(5,15);   <<PURCHASE ORDER NUMBER>>
    0013      ITEMN,    I1(5,5);    <<LINE ITEM ON ORDER   >>
    0014      QTY,      I1(5,5);    <<QUANTITY OF ITEM     >>
    0015      PARTN,    U16(5,5);   <<VENDOR PART-NUMBER   >>
    0016      DESC,     U16(5,5);   <<DESCRIPTION          >>
    0017      PRICE,    R2(5,5);    <<EXPECTED PRICE       >>
    0018      UNITM,    U4(5,5);    <<UNIT OF MEASURE      >>
    0019
    0019      <<  KEY INFORMATION                           >>
    0019      POKEY,    U4(5,15);   <<PURCHASE ORDER KEY   >>
    0020
```

```
SETS:
      NAME:  PUR01,A,CR015;  <<PO AUTOMATIC MASTER>>
      ENTRY:
             POKEY(2);
      CAPACITY:1001;

      NAME:  PUR02,D,CR015; <<PO HEADER DETAIL SET>>
      ENTRY:
             POHEAD(PUR01),   <<HEADER KEY              >>
             CDATE,
             VENNAM,
             VENADD,
             SHPWHS,
             SHPADD,
             DEPCOD,
             RDATE,
             CARRY,
             TERMS,
             AGENT;
      CAPACITY:  1001;

      NAME:  PUR03,D,CR015;  <<PO LINE ITEM DETAIL SET>>
      ENTRY:
             POITEM (PUR01), <<LINE ITEM KEY          >>
             ITEMN,
             QTY,
             PARTN,
             DESC,
             PRICE,
             UNITM;
      CAPACITY:5001;
END.
```

DATA SET NAME	TYPE	FLD CNT	PATH CNT	ENTR LGTH	MED REC	CAPAC CT	CART NO.
PUR01	A	1	2	2	9	1001	CR015
PUR02	D	11	1	83	3	1001	CR015
PUR03	D	7	1	24	3	5001	CR015

```
NUMBER OF ERROR MESSAGES   0
ITEM NAME COUNT:  19
DATA SET COUNT:  3
ROOT LENGTH:   2 BLOCKS, 226 WORDS
```

CARTRIDGE REFERENCE NUMBER	NUMBER OF BLOCKS REQ'D
CR015	1815.

OUTPUT PROCESSING

In our example above, it may turn out that purchase order input is not as important as being able to access the purchase order by various other organizations. Other organizaitons, however, probably may prefer to access the purchase order in another way, perhaps by vendor or part-number. In this case the source document, the purchase order, may still be used as a starting point for data base design, but additional information may have to be added to the data base to satisfy other departments' needs., For example, a purchasing agent may have a series of vendors to keep track of. The agent would like to know what orders the vendor currently has, but would like to keep track of other information such as delivery performance and total dollars for the year. This points to the inclusion of a vendor data set in the data base. Another example might be that an inventory control clerk needs to know what is on order for a certain part. If inventory is low on that part, it may be worth expediting the order. Figure C gives an example of a modified IMAGE schema for handling these additional requirements. The additional manual master for vendors allows an application program to handle both vendor information and all the orders currently open to that vendor. The additional manual master for part number allows an applications program to handle parts currently on order.

```
HEWLETT-PACKARD IMAGE/1000 DATA BASE DEFINITION PROCESSOR

$CONTROL LIST,NOROOT,NOSET,TABLE;
BEGIN DATA BASE PUR00;CR015;01;
LEVELS:
      5   CLERK;
      10  SUPER;
      15  MGR;
ITEMS:
    0001      <<   KEY INFORMATION                          >>
    0001      POKEY,   U4(5,15);   <<PURCHASE ORDER KEY     >>
    0002      VENKEY,  U24(5,15);  <<VENDOR NAME KEY        >>
    0003      PRTKEY,  U16(5,15);  <<PART-NUMBER KEY        >>
    0004
    00004     <<  VENDOR INFORMATION                        >>
    0004      VENADD,  U36(5,15);  <<VENDOR ADDRESS         >>
    0005      DELPER,  U4(5,15);   <<VENDOR DEL PERF CODE   >>
    0006      TOTDOL,  R2(5,15);   <<VENDOR TOTAL YR $$$    >>
    0007
    0007      <<  PART INFORMATION                          >>
    0007      PDESC,   U16(5,15);  <<PART DESCRIPTION       >>
    0008
    0008      <<  HEADER INFORMATION                        >>
    0008      POHEAD,  U4(5,15);   <<PURCHASE ORDER NUMBER  >>
    0009      CDATE,   U6(5,15);   <<PO CREATION DATE       >>
    0010      VENNAM,  U24(5,15);  <<VENDOR NAME            >>
    0011      SHPWHS,  U24(5,15);  <<SHIP TO WAREHOUSE      >>
    0012      SHPADD,  U24(5,15);  <<WAREHOUSE ADDRESS      >>
    0013      DEPCOD,  U6(5,15);   <<PURCHASE FOR DEPART    >>
    0014      RDATE,   U6(5,15);   <<REQUIRED DATE          >>
    0015      CARRY,   U12(5,15);  <<SHIPPING CARRIER       >>
    0016      TERMS,   U12(5,15);  <<SPECIAL TERMS          >>
    0017      AGENT,   U24(5,15);  <<PURCHASING AGENT       >>
    0018
    0018      <<  LINE ITEM INFORMATION                     >>
    0018      POITEM,  U4(5,15);   <<PURCHASE ORDER NUMBER  >>
    0019      ITEMN,   I1(5,5);    <<LINE ITEM ON ORDER     >>
    0020      QTY,     I1(5,5);    <<QUANTITY OF ITEM       >>
    0021      PARTN,   U16(5,15);  <<VENDOR PART-NUMBER     >>
    0022      DESC,    U16(5,5);   <<DESCRIPTION            >>
    0023      PRICE,   R2(5,5);    <<EXPECTED PRICE         >>
    0024      UNITM,   U4(5,5);    <<UNIT OF MEASURE        >>
    0025
SETS:
      NAME:PUR01,A,CR015;   <<PO AUTOMATIC MASTER >>
      ENTRY:
           POKEY(2);
      CAPACITY:1001;

      NAME:  PUR02,M,CR015;  <<VENDOR MASTER FILE>>
      ENTRY:
           VENKEY(1),
           VENADD,
           DELPER,
           TOTDOL;
      CAPACITY:503;
```

```
      NAME:  PUR03,M,CR015;  <<PART-NUMBER MASTER FILE>>
      ENTRY:
           PRTKEY(1),
           PDESC;
      CAPACITY:  6001;

      NAME:  PUR04,D,CR015;  <<PO HEADER DETAIL SET   >>
      ENTRY:
           POHEAD(PUR01),  <<HEADER KEY          >>
           CDATE,
           VENNAM(PUR02),  <<  VENDOR KEY        >>
           SHPWHS,
           SHPADD,
           DEPCOD,
           RDATE,
           CARRY,
           TERMS,
           AGENT;
      CAPACITY:  1001;

      NAME:  PUR05,D,CR015;  <<PO LINE ITEM DETAIL SET   >>
      ENTRY:
           POITEM(PUR01),  <<LINE ITEM KEY          >>
           ITEMN,
           QTY,
           PARTN(PUR03),
           DESC,
           PRICE,
           UNITM;
      CAPACITY:5001;
END.
```

DATA SET NAME	TYPE	FLD CNT	PATH CNT	ENTR LGTH	MED REC	CAPAC CT	CART NO.
PUR01	A	1	2	2	9	1001	CR015
PUR02	M	4	1	34	6	503	CR015
PUR03	M	2	1	16	6	6001	CR015
PUR04	D	10	2	71	5	1001	CR015
PUR05	D	7	2	24	5	5001	CR015

```
NUMBER OF ERROR MESSAGES   0
ITEM NAME COUNT:  24
DATA SET COUNT:  5
ROOT LENGTH:   3 BLOCKS, 296 WORDS

       CARTRIDGE REFERENCE NUMBER       NUMBER OF BLOCKS REQ'D

                 CR015                          3006.
```

CONCLUSION

The original source document, then, was used as a starting point to data base design. Further design information was added by talking to people who use the form and the information on the form. While the final data base design may not resemble the original paper form or report, it is one of the best places to start.

DEBUGGING IMAGE

By Todd Field

INTRODUCTION

This article is designed to help find IMAGE/1000 bugs. It concentrates on the type of problem that can be produced through a known procedure rather than a problem which occurs at random. However, the process of defining the procedure itself is a major step in the debugging process.

THE SYMPTOMS

- The data has been entered. The user runs QUERY to verify the data has been entered. A chained read (i.e., FIND keyval IS "xxx" END;) fails to turn up the data. However, a serial read (ie FIND keyval ILT "yyy" AND keyval IGT "www" END;) turns it up. Furthermore, this was a data item which was entered last week. This week's entries all look ok.

- The chained read finds it alright, but the wrong entry is in the data record.

- Empty records are in the same chain with non-empty records.

- DBSPA shows a corrupt data base. This indicates that the free list count does not agree with the actual number of empty records.

- Generally speaking, your pointers and chains are so mangled that it looks like every elephant ever owned by Barnum and Bailey Circus sat on your 7920.

A data base is said to be corrupt whenever the data in it is not present in the correct form, is not in the correct location, or is not linked together properly. The data is not linked properly if a forward or backward pointer from one data entry in a detail data set does not point to the next entry in the chain, if a forward or backward pointer in a synonym chain in a master data set is incorrect, if a pointer from an entry in a master data set does not point to a corresponding entry in a detail data set, or if the free list (list of empty records) is incorrectly threaded. For an explanation of some of these terms, see ***Gary McCarney's*** article, "An Introduction to Data Base Management Terminology" in issue 16 of the Communicator/1000.

DETECTING THE PROBLEM

Assume you have a good data base (not exhibiting any sign of corruption as described above). If you are fortunate enough to be present at a before=good, after=bad session, you have the problem licked. Grab the program that was running, a hard copy of the data base and a list of the other programs running at the time and go directly to GO. Since you have the program which is causing the data base corruption, you can tell, by examining it or with some help from your local HP Systems Engineer (SE), where the data is going bad. However, in most cases, you will not be so lucky. Usually all you will get to see is the data base after it has been corrupted.

After the problem has occurred several times, you should start to ask some of the following questions. They will not necessarily give you all the information you need to solve the problem, but hopefully, you will be able to locate areas of further exploration from the answers.

- Revision code? Be sure that you are working the the most recent revision of RTE and of IMAGE/1000.

- Are there any known IMAGE problems? Check the Software Status Bulletins (SSB) for reported IMAGE bugs.

- When did you last know the data base was good? This may not be obvious, as the last several times the user accessed the data base the offending records may not have been accessed.

- In what mode was the data base open? If it was mode 1 open, you should look elsewhere for the problem.
- What else was going on in the system at the time?
- How many programs access the data base at one time? Does the problem re-occur if only one program has the data base open?
- If mode 2 open, what else in your system uses System Available Memory (SAM)?
- Do the programs perform bounds checks on all array references? Nothing causes problems like zeroing out the rest of the partition.
- Did the program terminate abnormally in any way? This includes an operator "OFing" the program.
- What kind of error checking do the programs do? How do they terminate. Have the programs ever been aborted or terminated incorrectly? A frequent cause of IMAGE problems is in improperly terminating programs.

AVOIDING THE PROBLEM

The most obvious cause of IMAGE problems is a program terminating without closing the data base properly. If the program is open in mode 3 and terminates after doing DBPUT's or DBDEL's without closing the data base, all bets are off and you deserve whatever you get. Mode 2 is not as bad, but a little effort needs to be put forth to save the data base.

The reason an improper mode 3 termination is so disastrous is not difficult to understand. In mode 3, the run table is stored in the partition with the program. The run table is a copy of the root file which contains information such as set and item names, path descriptions and set names. It also contains more volatile information, such as access mode and current position in chains. Also stored in the partition are the Data Control Blocks (DCBs) for the various data sets. DBPUTs and DBDELs alter the run table and these DCBs, but the data in the DCBs are not written to the data set until more data is needed from the data set (chain maintenance is an example) or until a DBCLS is done. DBCLS is the only method to post the run table to the root file. Thus, if the program abnormally terminates, the run table will be lost (affecting free list pointers) and the data in the DCB's will be lost (probably affecting chain pointers).

Mode 2 handles the run table and the DCBs in a different fashion. As well as being stored in the partition, the free record pointers for each data set are written into SAM through class I/O, to be shared by all other mode 2 programs now using this data base. The DCBs are still stored in the partition. The other difference is that the data in the DCBs is posted and an updated run table is written to SAM after each DBUNL call. Thus, if the program abnormally terminates after a DBUNL, the data in the data sets will be correct but the free record information will still be in SAM where it can be posted using RECOV. If DBUNL has not been called, the results are the same as in a mode 3 open. Correct program termination cannot be overemphasized.

Some things to avoid are:

- Allowing your operators to abort IMAGE programs. In the worst case, they can suspend the program.
- Reporting an error and PAUSEing. (Does your operator know how to use the GO command?)
- Infinite loops in error routines. Always leave a way out of an input cycle. For example, if /A is entered, jump to a routine to terminate processing in an orderly fashion.
- Do not write all over computer memory. IMAGE use the area between the end of your program and the end of your partition to work in. Accidentally zeroing out one of IMAGE's DCBs is a fast way to corrupt any file.
- Do not alter the media record. The media record consists of the first several words of each record, before the user data. This data contains chain linking information.

Some things to do are:

- Check the error return parameter from IMAGE subroutine calls, IERR, after every data base access. If you get an error on any subroutine, report it and take some action. If an error develops, it falls into one of three categories.

  a. Item not present in an interactive program. Prompt for the item again and check for an "abort" answer. Consider closing the data base while waiting for more input.

  b. Hard error (i.e., DBPUT fail). Jump immediately to an error ending routine.

  c. Non-IMAGE error (user file read error, etc.). Treat these error as you would treat a. or b. above.

- Use the "no abort" bit on all EXEC calls.

If you think that abnormal program termination may be your problem and you are now using a mode 3 open, consider using a mode 2 open. If you want to, try calling DBLCK/DBULK as often as possible. Now only your free record information will be left in memory if the program is abnormally terminated, and RECOV may be able to post this to the root file. RECOV can recover the run table even if the data base was locked when the program terminated. This is possible because the resource number associated with the lock was acquired globally (meaning that anyone can release it) and locked locally (meaning that it is unlocked automatically upon program termination). This is not a foolproof method to solve the problem, but it may give you enough breathing room to work in.

RECOVERING FROM THE PROBLEM

At some point your data base may contain bad or corrupt data and need to be reloaded or otherwise fixed. This is almost always a time consuming process. As this article is geared toward frequently occuring problems, a short discussion of various recovery techniques is in order.

RECOV is an HP supplied utility that retrieves the run table from SAM and posts it to the root file. If you had the data base open in mode 2 and if you had locked/unlocked the data base recently, RECOV may be able to restore all your data. Run DBSPA immediately after RECOV to be sure the free list count is the same. Remember though, DBSPA can run successfully and you could still have a corrupt data base. The IMAGE/1000 manual (92063-90001) discusses RECOV in more detail.

Even if your chains are bad, the data in your data base may still be good. The reason for this is that IMAGE uses a fixed precedence when it updates a record. IMAGE updates, in order

The root file (kept in memory for the duration of each program)
Master set chains
Detail set chains
Detail records

Thus, it is possible (probable) that the root file, the last detail record and the last chain maintenance in each set will be lost if the program abnormally terminates. If a large amount of data has been entered, it may be acceptable to lose this data rather than reenter all data since the last backup tape.

If you have entered a large amount of data since your last backup and your pointers are badly fouled up, dumping and reloading the data base using DBULD and DBLOD will allow you to recover most of the data. DBULD reads the data base serially and records the user data, but not the media record, on the tape. Thus, DBLOD will create a good data base even if the original data base was corrupt. Unfortunately, this may take a long time. See the IMAGE/1000 manual for more detail.

DBULD and DBLOD allow the user the option of redefining the data base schema. Consider writing your own program to perform the same function as DBULD and DBLOD, but without the option of redefining the schema. A user-written program to execute one specific function will run faster than general purpose programs such as DBULD and DBLOD.

TRACKING DOWN THE PROBLEM

After you have observed a corrupt data base, you still have to track the problem down. Below are several tools to help you do this.

WALKING THROUGH CHAINS

The easiest way to really prove that a data base is corrupt and to understand where and why it is corrupt is to walk through the data sets manually. To do this you need to know:

1. What you are looking at
2. What you are looking for
3. How you look for it

What you are looking at:

Each record in the data set contains not only the user data, but also the media record containing the chain pointers and an empty/occupied flag. The format is different for master and detail data sets. Both master and detail data sets are FMP files with the file name the same as the data set name and the security code the negative of the data base security code. Note that only the first and last paths are listed below. The other paths have the same format. Figures 1 and 2 show the formats for records in master and detail data sets.

What you are looking for:

For any suspicious record in the data base, find the record and follow its synonyms, forward and backward pointers and links to other data sets until you find something suspicious, such as pointing to a free record, synonyms incorrect, forward and backward pointers not matching or any other peculiar looking circumstance. If you find something, try to decide how or why it was corrupted. Is it in another record's chain? Was chain maintenance being done for a new or deleted record in the chain? When were the adjacent records added or deleted? This kind of debugging is no different than any other kind; it takes practice.

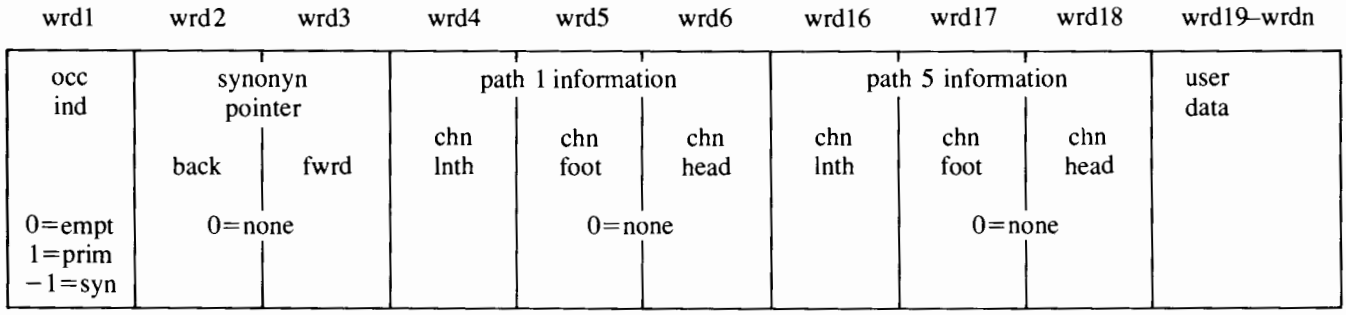
How to look for it:

The easiest way to follow a chain is to dump out the entire data base in octal. This is fine if you have a rather small data base, but for any practical application, it would take quite a bit of paper. A little known fact is that FMGR's LIst command can be used to print out individual records in a file. To do this try

```
:LI,filename:-sc:crn,B,[starting record],[ending record]
```

Although this will only list out the first 128 words of the record, this can be very useful. The media record is contained in the first several words of the record, and the remainder of the 128 words will usually give you enough information to follow any chains in this file.

A good way to find a starting point is to use QUERY to find a suspect record and list out the select file in octal. Each entry in the select file (up to the number of records found) is the relative record number of an entry in a data set. Remember to convert binary to decimal in the appropriate places. (FMGR can do this for you. Simply enter numbers in FMGR commands followed by B to convert the number from octal. For example, 17 octal should be entered as 17B.)



Occ ind is the occupational indicator (non-zero if there is anything in this record).

Figure 1. Master Data Set Record

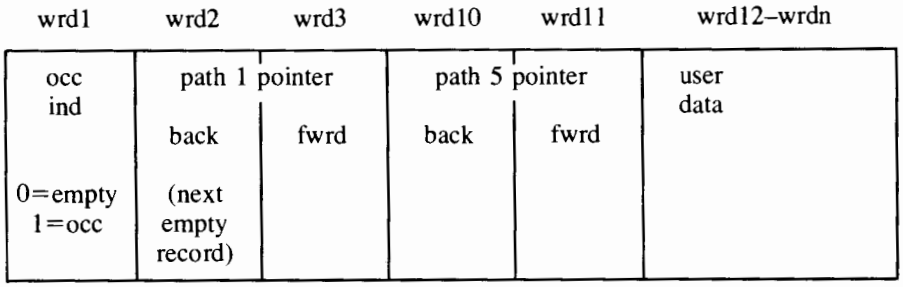


Figure 2. Detail Data Set Record

THE ROOT FILE

If DBSPA shows a discrepancy between the number of empty records in the data base and the number of empty records that IMAGE think are in the data base, it is likely that your root file has been corrupted. Figures 3 through 8 show the format of the root file and related entries. Note that the record length of the root file is greater than 128 words, so you will be unable to use the FMGR DU or LI commands to interrogate the root file. Note that entries marked with asterisks (**) are filled into the run table at run time. The root file is named after the data base name and is located on the disc cartridge specified in the schema. Its security code is also the negative of the data base security code. Figure 3 shows the format of the root file, figure 4 the item table entries, and figure 5, the set table entries.

WORD	ROOT FILE (RUN TABLE)	
1		
2		
3		
4	"L"	"B"
5	DATA BASE SECURITY CODE	
6	DATA ITEM COUNT	
7	DATA SET COUNT	
8	POINTER TO DATA ITEM TABLE	
9	POINTER TO DATA SET TABLE	
10	LEVEL	MODE
11	LEVEL	
12	ONE	
13	WORD	
	. . .	
53	LEVEL	
54	FIFTEEN	
55	WORD	
	ITEM TABLE ENTRIES (ONE ENTRY PER ITEM)	
	SET TABLE ENTRIES (ONE ENTRY PER SET)	

**

Figure 3. Root File Format

WORD	ITEM TABLE ENTRIES	
1	DATA	
2	ITEM	
3	NAME	
4	READ LEVEL	WRITE LEVEL
5	ITEM TYPE	DATA SET #

type is I, R or U

Figure 4. Item Table Format

WORD	SET TABLE ENTRIES		
1	0	DATA SET TYPE	type is D, M or A
2	MEDIA RECORD LENGTH		
3	ENTRY LENGTH		
4	ITEM COUNT	PATH COUNT	
5	SEARCH ITEM #	PATH #	** changes only in detail
6	POINTER TO PATH TABLE		
7	FREE RECORD COUNT		
8	FIRST FREE RECORD		** always 0 for master
9	LAST ACCESSED RECORD #		**
10	PATH LENGTH OF CURRENT CHAIN		**
11	RECORD # OF CURRENT CHAIN FOOT		**
12	NEXT RECORD # IN CHAIN		**
13	DATA		
14	SET NAME		
15		CARTRIDGE #	
16	CAPACITY		
	RECORD DEFINITION ENTRIES (ONE PER ITEM)		
	PATH TABLE ENTRIES (ONE PER PATH)		

Figure 5. Set Table Format

WORD	RECORD DEFINITION ENTRIES			
1	ITEM #	R	W	LENGTH (W)

Figure 6. Record Definition Entry Format

WORD	PATH TABLE ENTRIES (MASTER)		
1	SEARCH ITEM #	DATA SET #	# refers to related detail.
2	PATH #	SCRATCH	SCRATCH flag is set to delete, update or add item when a DBPUT or DBDEL is done to the related detail.

Figure 7. Master Path Table Entry Format

WORD	PATH TABLE ENTRIES (DETAIL)		
1	SRCH ITM INDEX	DATA SET #	INDEX of items in this set.
2	PATH #	SCRATCH	# refers to related master.

Figure 8. Detail Path Table Entry Format

TRACE: DECIMAL TO OCTAL AND HASHING ROUTINE

As an assist in trying to follow chain pointers and hash key values, appendix 1 contains the listing of a simple FORTRAN program, TRACE. TRACE asks you for a value in decimal or octal and prints it out in octal and decimal. If you enter a value of H, TRACE will prompt you for a key to a master data set and information about the key (integer or ASCII, etc.) TRACE will then call the hashing function of IMAGE directly and print out the hashed value modulo the number of entries in the data set. For a further description of hashing, please read *Gary McCarney's* article mentioned above.

CONCLUSION

One certainly hopes that one will not need to delve into a data base in as much detail as is presented here. The same tools and techniques, however, can be used for other purposes. Application program debugging and fine tuning an application can be difficult without a detailed knowledge of the internal workings of the system being worked with. I hope that this article will enable you not only to track down possible system bugs, but also to form a better fit between your applications and HP/1000 systems.

APPENDIX 1: TRACE ROUTINE LISTING

```
FTN4,L
C
C     PROGRAM TO DO OCTAL/DECIMAL CONVERSION OR HASH AN IMAGE KEY VALUE
C
C     WRITTEN BY:   TODD FIELD      MARCH, 1978
C
      PROGRAM TRACE
C
      INTEGER IPARM(5),IBUF1(33),IBUF2(33)
      EQUIVALENCE (RBUF,IBUF1(1))
C
      CALL RMPAR(IPARM)
      IF (IPARM(1).EQ.0) IPARM(1)=1
C
C     START LOOP
C
 10   WRITE (IPARM(1),20)
 20   FORMAT(/" INPUT A DEC OR OCT NUMBER (0 TO STOP H TO HASH) ? _")
      READ(IPARM(1),30) (IBUF1(I),I=1,33)
 30   FORMAT(33A2)
      IBUF1(33)=2H,
      ICCNT=33
      CALL PARSE(IBUF1,ICCNT,IBUF2)
C
C     DECIDE WHETHER TO HASH OR NOT
C
      IF (IBUF2(1).EQ.1)  GOTO 100
C
C     HASH
C
      WRITE (IPARM(1),40)
 40   FORMAT(/" MODE (0=ASCII, 1=INT, 2=REAL) ? _")
      READ (IPARM(1),*) IMODE
      WRITE (IPARM(1),41)
 41   FORMAT(" VALUE ? _")
      GOTO (50,60,70) IMODE+1
```

```
C
C     ASCII
C
 50   READ (IPARM(1),55) (IBUF1(I),I=1,33)
 55   FORMAT(33A2)
      WRITE (IPARM(1),56)
 56   FORMAT(" NUMBER OF CHARACTERS TO HASH ? _")
      READ (IPARM(1),*) ILEN
      GOTO 80
C
C
      INTEGER
C
 60   READ (IPARM(1),*)  IBUF1(1)
      ILEN=1
      GOTO 80
C
C     REAL
C
 70   READ (IPARM(1),*)  RBUF
      ILEN=2
C
C     HASH
C
 80   CALL HASH(IBUF1,ILEN)
      CALL ABREG(IPTR,IBDUM)
      WRITE (IPARM(1),85)
 85   FORMAT(" NUMBER OF ITEMS IN THE DATA SET ? _")
      READ (IPARM(1),*) ITEMS
      IPTR=IPTR+1
 90   IF (IPTR.GT.ITEMS) IPTR=IPTR-ITEMS
      IF (IPTR.GT.ITEMS) GOTO 90
      GOTO 105
C
C     SIMPLE INPUT
C
 100  IPTR=IBUF2(2)
C
C     OUTPUT VALUE
C
 105  WRITE (IPARM(1),110) IPTR,IPTR
 110  FORMAT(" DECIMAL:  ",I6,"   OCTAL:  ",K6)
      IF (IPTR.NE.0) GOTO 10
      END
```

Samantha hasn't received any questions from readers this month. However, in going through the papers on her desk, she found the following.

"Dear Samantha,

I'm confused. When should I be using a DBCLS mode 1 (post mode) and where should I put it?

Sincerely,
Ima L. Ostchain"

Dear L. Ostchain,

DBCLS in post mode should be used when you have the data base open in mode 2 (shared read/write access), you have locked the data base, done a DBPUT or a DBDEL, you are about to unlock the data base and you want an additional level of security protecting you from a system crash. DBCLS in post mode simply takes the free list pointers which IMAGE stores in System Available Memory and posts them to the disc. The correct calling sequence should be:

```
C       PROCESSING
        CALL DBINT(IBASE,ISCOD,ILIST,ISTAT)
        IF(ISTAT.NE.0)GO TO 9000
        CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
        IF(ISTAT.NE.0)GO TO 9000
C       PROCESSING
        CALL DBPUT(IDSET,ISTAT,INBR,IVALU,IBUF)
C       OR
        CALL DBDEL(IDSET,ISTAT)
        IF(ISTAT.NE.0)GO TO 9000
C       PROCESSING
        CALL DBCLS(1,ISTAT)
        IF(ISTAT.NE.0)GO TO 9000
        CALL DBULK(ISTAT)
        IF(ISTAT.NE.0)GO TO 9000
```

```
C     PROCESSING
P     WRITE(LOGLU,8999)
8999  FORMAT("SUCCESSFUL TERMINATION!")
      GO TO 9500
9000  CONTINUE
      WRITE(LOGLU,9001)ISTAT
9001  FORMAT("UNSUCCESSFUL TERMINATION!")
      CALL DBULK(ISTAT)
9500  CALL DBCLS(0,ISTAT)
      IF(ISTAT.EQ.0)GO TO 9999
      WRITE(LOGLU,9501)ISTAT
9501  FORMAT("UNSUCCESSFUL DBCLS!!! ISTAT IS: ",I6)
9999  CONTINUE
      END
```

For more information on free list pointers, SAM and what happens when an IMAGE data base is open in mode 2, read the article "Debugging IMAGE" in this issue of the COMMUNICATOR.

Do you do a lot of tangent calculations? Then ***Larry B. Smith*** of Hewlett-Packard in Albuquerque has a tip for you.

"Here's a suggestion for those of you that calculate trigonometric functions in your RTE system. The TAN subroutine in the RTE Relocatable Library is used to calculate the tangent of a real X, where X is in radians, i.e.,

```
Y=TAN(X)
```

Did you know that it's faster to calculate the tangent of an angle this way?

```
Y=SIN(X)/COS(X)
```

Although this method seems slower because RTE calls the SIN subroutine, then the COS subroutine, and then divides the two answers to get the final result, it's actually much faster. It's not unusual to achieve 40-60% speed improvement using this method.

If you are using the TAN subroutine and speed is critical to your application, try this suggestion.

It's also worth noting that because we don't have a double precision tangent subroutine in the RTE library, this method works fine using DSIN and DCOS subroutines, i.e.,

```
DOUBLE PRECISION X,Y
Y=0.2
X=DSIN(Y)/DCOS(Y)
```

Try it, you'll like it.

Regards,
Larry B. Smith"

Software Samantha is here to answer your questions about HP1000 software. If you have any problems, questions, points that need clarification, Samantha is glad to help. All letters will be answered, whether or not they are printed in the COMMUNICATOR. Address your letters to

SOFTWARE SAMANTHA
c/o Editor, COMMUNICATOR
Hewlett Packard Data Systems
11000 Wolfe Road
Cupertino, California 95014

NEW FEATURES ARE ADDED TO BASIC/1000D

by Van Diehl

Several new important features have been added to BASIC/1000D (92101A). The most important is the addition of formatted output capability, via the PRINT USING statement.

Formatted output, with PRINT USING, gives you a simple way to generate reports, as shown in the example below.

PRINT USING statement specifies the format that the variables specified in the statement are to be printed. This format can be in a literal string, in a string variable, or in a special statement called the IMAGE statement. In contrast, the standard PRINT statement does not allow the specification of a format.

With PRINT USING the user has explicit and exact control over the format of the program output:

- Numbers can be printed in three different representations: integers, fixed point and floating point.
- The exact position of plus and minus sign can be specified
- String values can be printed in specified fields and literal strings and blanks can be inserted wherever needed.
- Full control over carriage returns and line feeds is possible.
- Arbitrary long lines can be printed without the carriage returns and line feeds normally provided by the PRINT statement.

Another important addition to BASIC/1000D is the INVOKE statement. The INVOKE statement complements the CHAIN statement. The INVOKE statement is used to schedule a second BASIC program from a calling program. The called program may also call another program and so on. When the current executing, called program, terminates, control is returned to the calling BASIC program. BASIC data files remain open when one program INVOKEs another and any TRAPs previously enabled will remain enabled.

The CHAIN statement instead terminates the current executing program and begins execution of another program. No files are left open or TRAPs enabled.

These features now complement the extensive set that is provided by BASIC/1000D, such as:

- Multi-user Operation
- Disc file access for programs and data
- BASIC/1000D can call FORTRAN, ASSEMBLER or microassembler coded subroutines.
- Commands for tracing, setting breakpoints and simulating subroutine calls to non-implemented subroutines
- Character string variables
- CHAIN and INVOKE statements to implement larger programs (larger than the 1500 statements (ballpark number) that can be loaded in a RTE-IV partition).
- FORMATTED OUTPUT with PRINT USING
- Interface call to IMAGE/1000
- Decimal string arithmetic
- Instrumentation interface
- Time scheduling of BASIC tasks via START and TRNON statements
- Event scheduling of BASIC tasks via TRAP statement
- Bit manipulation statements for setting, clearing, shifting and performing boolean arithmetic
- Interface to GRAPHICS/1000, allowing easy access to the 2648 graphics terminal, 7245 plotter-printer, and 9872 graphics plotter

HOW DO CUSTOMERS GET THE NEW ENHANCED BASIC/1000D?

All subscribers of the Comprehensive Software Support and Software Subscription Service will be (or have been already) updated, with the revision 1826 of the BASIC/1000D software. Customers that do not subscribe to these services can get the enhanced version of BASIC by purchasing the 92101A product. See your local Hewlett- Packard sales representative for more information.

REPORT GENERATION EXAMPLE

This program is a sample report generator. It first requests a school number from the terminal, then reads and prints out information about the school's teachers from a file. Note that a carriage control character is used to advantage (statement 100), slashes (/) are used (statement 200) string and fixed-point fields are used (statement 210), and an error occurs in the output for the eighth teacher (number too large for field; therefore, it is printed in E format on a separate line).

```
10 REM THIS PROGRAM GENERATES A REPORT ON TEACHERS
50 DIM A$(25),B$(19),C$(19)
60 FILES SCH1,SHC3,SHC3,SCH4,SCH5
100 IMAGE #,"ENTER SCHOOL NUMBER:"
150 IMAGE "TEACHER",13X,"SUBJECT",13X,"SALARY",4X,"ATTND."
175 IMAGE "-------",13X,"-------",13X,"------",4X,"------"
200 IMAGE "CENTRAL CITY SCHOOL DISTRICT"/"DAILY REPORT OF ",25A//
210 IMAGE 20A,20A,"$",DDD.DD,DD.DDDD
230 PRINT USING 100
250 INPUT Z
260 READ #Z;A$,N
270 PRINT
500 PRINT USING 200;A$
550 PRINT USING 150
555 PRINT USING 175
557 FOR A1=1 TO N
560 READ Z;B$,C$,A,B
600 PRINT USING 210;B$,C$,A,TAB(50),B
620 NEXT A1
1000 END

ENTER SCHOOL NUMBER:?1

CENTRAL CITY SCHOOL DISTRICT
DAILY REPORT OF B. BAKER HIGH SCHOOL

   TEACHER              SUBJECT              SALARY       ATTND.

   MISS BROOKS          ENGLISH              $450.34      12.5000
   MISS CRABTREE        REM. READING         $400.00      64.3200
   MISS GRUNDIE         HISTORY              $350.00      1.0010
   MRS. HUMPREY         SPELLING             $700.00      99.9900
   COLONEL MUSTARD      CRIMINOLOGY          $700.00      21.4500
   MISS PEACH           LIFE PREPARATION     $232.00      23.2320

   PROF. PLUM           AGRICULTURE          $777.77      65.0050
   MISS H. PRYNNE       SOCIAL STUDIES       $100.25
   +5.00500E+02

   MISS SCARLETT        P.E.                 $205.10      25.0000
   MR.SIR               HOME ROOM            $890.00      99.9000
   MR. T. TIM           MUSIC                $ 10.99      0.0500
   MR. WEATHERBY        ECONOMICS            $767.99      10.0400
```

RTE-II/III to RTE-IV UPGRADE COURSE AVAILABLE

If you are one of the many customers who are planning to upgrade your existing RTE-II or RTE-III Operating System installation to the new RTE-IV Operating System, take note: A two day *RTE-II/III to RTE-IV Upgrade Course* is available. This course, which assumes a thorough knowledge of RTE-II/III as a prerequisite, will provide you with detailed information on all of the new features of RTE-IV. Class time is divided between lecture material which explains the new features, and hands-on lab time with the RTE-IV Operating System. Also supplied is a complete set of new manuals, such as the RTE-IV Programming and Reference Manual and the RTE-IV Generation Manual. Course fee is $250.00 in the United States. Contact your local HP representative for a course data sheet and the current schedule of classes.

SETTING UP A TRAINING PROGRAM

We encourage you to discuss your training requirements with your local HP representative. This person is trained to assist you in setting up an optimum training plan for your needs. However, the following comments about the HP 1000 Computer Systems training program may help you to prepare in advance for this discussion.

In general, courses should be taken in the sequence indicated in the training program diagram on the next page, starting from the left, and proceeding toward the right. Completion of each course in sequence will ensure that all needed prerequisites are satisfied.

If you have not had any previous experience with minicomputer systems, you should start your training with the four day *Introduction to HP Minicomputers course*. Otherwise, you can skip this course, and begin your training with either the *HP 1000 Disc-Based* or *Memory-Based RTE Operating System* course. Which one you choose will depend upon the type of system in your installation. Note however, that both of these courses require a thorough knowledge of FORTRAN programming as a prerequisite.

All HP 1000 Computer System users should plan to take one of the Operating Systems Courses described above. Further training is optional, depending upon the nature of your programming tasks. For example, if you are planning to:

- Design a data base using the IMAGE/1000 software. . .
  You should take the *IMAGE/1000 Data Base Management* course (22977A).

- Connect instruments to your HP 1000 via the HP-IB. . .
  You should take the *HP-IB in a Minicomputer Environment* course (22980C).

- Operate your system as part of a distributed systems (DS/1000) network. . .
  You should take the *DS/1000 User's* course (22987A). Furthermore, if you are to be designated as the Network Manager for your DS/1000 network, you should follow this course with the *Theory of Operation of DS/1000* course (22961B). And if your network will include an HP 3000 system, you should continue your training and take the one day *Theory of Operation for DS/1000 to HP 3000* course (22962B).

- Write programs in HP Assembly Language. . .
  You should take the *HP 1000 Assembler Programming* course (22952B). (Note that this course is a prerequisite for the Driver Writing and Microprogramming courses mentioned below.)

- Interface your own peripheral equipment to your HP 1000 system. . .
  You should take the *HP 1000 Driver Writing* course (22990A) to learn how to write device drivers for your own peripherals.

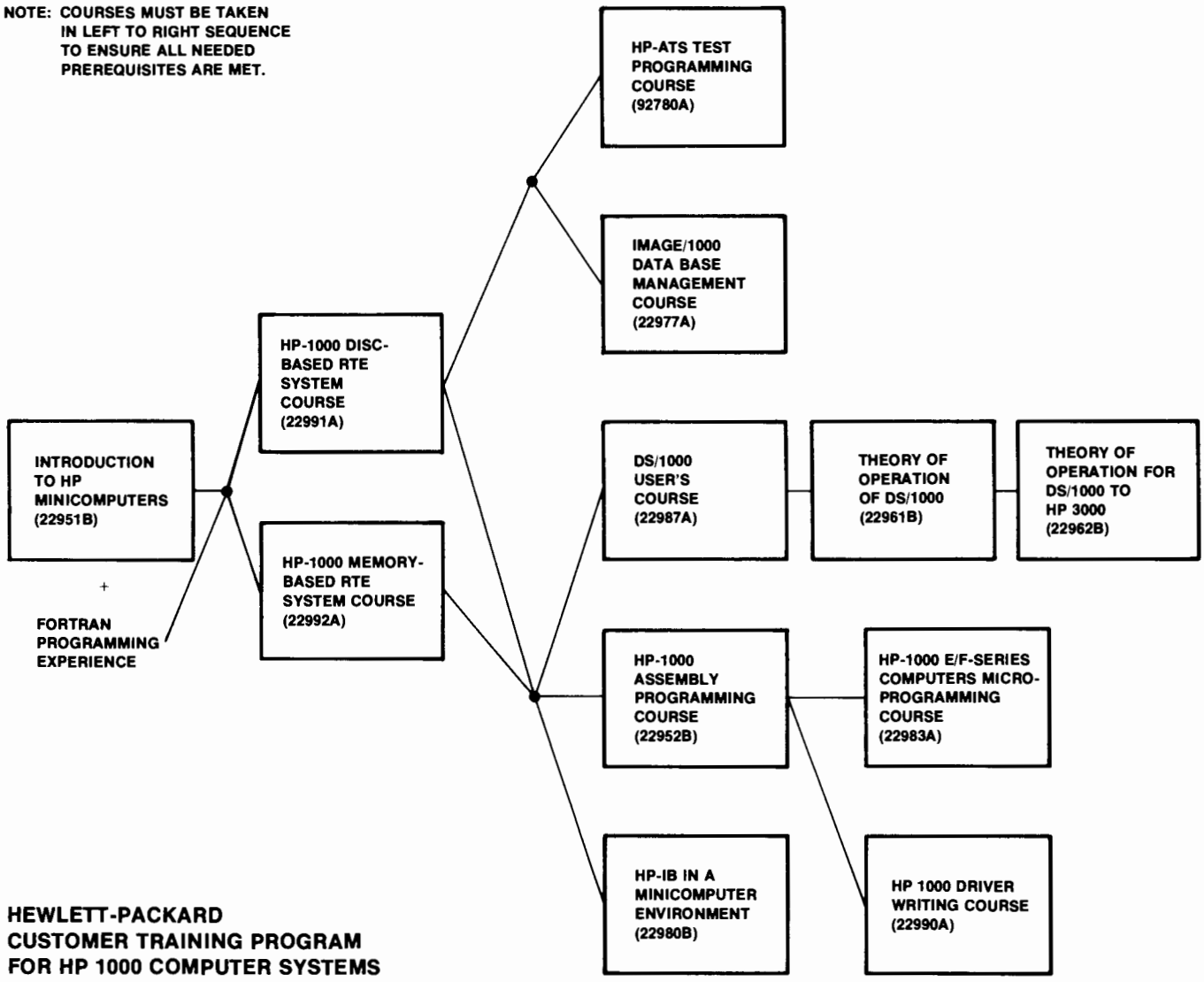
- Customize your computer for your application using the Microprogramming feature of the HP 1000. . .
  You should take the *HP 1000 E/F Series Computers Microprogramming* course (22983B).

- Write test programs for your HP-ATS system. . .
  You should take the *HP-ATS Test Programming* course (92780A).

SUMMARY

After reviewing the new customer training program discussed in this section, choose a tentative training plan that satisfies your needs. Then discuss your plan with your local HP representative. This person can assist you with your course selection, provide you with the latest course schedule, and register you in the appropriate courses at the nearest customer training center.

See you in class!



NEW COURSES

In the last issue of the COMMUNICATOR 1000, the new training program for HP 1000 computer systems was discussed. The introduction of this new program coincided with the introduction of the new RTE-IV operating system and the new HP 1000 F-series computers. A schedule for most of these courses is now available and is given later in this section. A brief description of each of the new and revised courses is given below. For more detailed information, refer to the data sheet for each course or contact your local HP representative.

22951B INTRODUCTION TO HP MINICOMPUTERS

Description: This course provides an entry point into HP computer training for those customers who have had no previous experience with minicomputer systems. Upon completion of the course, the student will be familiar with the concepts of:

1. HP minicomputer architecture.
2. Operating systems.
3. High level languages.

Length: 4 days

Lab: Provides a hands-on introduction to the hardware and software operation of HP 1000 minicomputers. This includes operation of the computer front panel, system boot-up procedures and on-line loading and execution of programs.

Prerequisites: None. Students may be either hardware or software oriented.

22991A HP 1000 DISC-BASED RTE SYSTEM

Description: This course covers the operation of the RTE-IV operating system in an HP 1000 system environment. This includes program preparation using standard compiler, assembler,editor, and loader; disc usage; system software generation; and use of the Batch-Spool Monitor (BSM), including the file manager. HP 1000 software support policies are also briefly discussed.

Length: 10 days

Lab: Provides hands-on experience in operating, programming, and generating the RTE-IV system, including BSM.

Prerequisites: Demonstrated proficiency in FORTRAN programming (such as completion of a FORTRAN programming course) and completion of the Introduction to HP Minicomputers course (22951B) or equivalent minicomputer experience.

REVISED COURSES

22980C HP-IB IN A MINICOMPUTER ENVIRONMENT

Nature of Changes: The material in the HP-IB course has been expanded to include programming information on the twelve new HP-IB message subroutines and information on how to write device subroutines for specific HP-IB devices. Device subroutines can simplify the use of HP-IB devices by programs by providing an easy-to-use software interface tailored to each device.

Description: This course provides an introduction to HP-IB concepts and theory as they apply to use in HP 1000 Computer System controlled measurement systems as well as training in the programming of HP-IB on an RTE system. Information on how to write device subroutines is also included.

Length: 4 days

Lab: Provides hands-on experience with a typical HP 1000 computer controlled HP-IB instrument system.

Prerequisite: Completion of either the HP 1000 Disc-Based RTE System course (22991A) or the HP 1000 Memory-Based RTE System course (22992A) or equivalent RTE experience.

22983B HP 1000 E/F-SERIES COMPUTERS MICROPROGRAMMING

Nature of change: This course has been improved and updated to include microprogramming information related to the recently introduced HP 1000 F-series computers. Information on E-series microprogramming is also included.

Description: This course covers the theory and use of HP microprogramming hardware and software to prepare, alter, and install microprograms for HP 1000 E- and F-series computers.

Length: 5 days

Lab: Provides hands-on experience with preparation and installation of microprograms.

Prerequisites: Completion of either the HP 1000 Disc-Based RTE System course (22991A) or the HP 1000 Memory-Based RTE System course (22992A) and the HP 1000 Assembler Programming course (22952B) or equivalent RTE and assembly language experience.

TRAINING SCHEDULE

The current schedule for customer training courses on HP 1000 computer systems products is given in this section. Included are courses offered both in U.S. and in Europe during the upcoming months.

You can also obtain a copy of the training schedule from your local HP sales office. A European course schedule is available through the sales offices in Europe; a U.S. schedule through U.S. sales offices.

*Prices quoted are for courses at the U.S. training centers only. For prices of courses at European training centers please consult your local HP sales office.

DATA SHEETS

Data sheets giving detailed information on each of the courses scheduled are available from your local HP representative.

REGISTRATION

Requests for enrollment in any of the above courses should be made through your local HP representative. That person will supply the Training Registrar at the appropriate location with the course number, dates, and requested motel reservations. Enrollments are acknowledged by a written confirmation indicating the training course, time of class, location and accommodations reserved.

ACCOMMODATIONS

Students provide their own transportation meals and lodging. The Training Registrar will be pleased to assist in securing motel reservations at the time of registration.

CANCELLATIONS

In the event you are unable to attend a class for which you are registered, please notify the Training Center Registrar immediately in order that we may offer your seat to another student.

U. S. TRAINING CENTER SCHEDULES, LOCATIONS, AND RATES

Course Number	Title	Length	Price	CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Cupertino)	Boise Division (Boise)
22951B	Intro to HP mini's	4 days	400	Dec 4	Sep 11 Oct 16 Nov 27 Jan 8	Sep 18 Oct 16 Nov 27 Jan 8				
22991A*	HP 1000 DISC RTE (Course includes RTE-IV operating system, batch spool monitor and file manager.)	10 days	1000	Sep 11 Sep 25 Oct 9 Oct 23 Nov 6 Nov 27 Dec 11 Jan 8 Jan 22 Feb 5 Feb 19	Sep 18 Oct 23 Dec 4 Jan 15 Feb 5	Sep 11 Sep 25 Oct 9 Oct 23 Nov 6 Nov 27 Jan 15 Jan 29 Feb 26				
22992A*	HP 1000 Memory RTE	10 days	1000	Dec 11		Dec 4				
22977A*	IMAGE	5 days	500	Sep 25 Nov 27 Jan 22	Oct 2 Dec 18 Feb 26	Nov 13 Feb 5				
22952B*	1000 ASMB	5 days	500	Oct 23 Dec 11 Feb 5	Nov 13 Jan 29	Oct 9 Nov 6 Jan 29				
22987A*	DS/1000 User's Course	5 days	500	Sep 18 Nov 6 Dec 11 Jan 29		Oct 23 Jan 8				
22961B*	DS/1000 Theory of Op.	4 days	400			Jan 15				
22962B*	DS/1000 to HP 3000 Theory of Op.	1 day	100			Jan 19				
22990A*	RTE-Driver Writing	3 days	300	Sep 11 Nov 6 Feb 19		Sep 6 Dec 18 Feb 21				

*These courses carry prerequisites — refer to the data sheet for each course for more information.

U. S. TRAINING CENTER SCHEDULES, LOCATIONS, AND RATES (Continued)

Course Number	Title	Length	Price	CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Cupertino)	Boise Division (Boise)
22980C*	HP-IB Minicomputer Environment	4 days	400	Oct 30 Jan 15						
22983B*	HP 1000 E/F Microprogram-ming	5 days	500	Oct 9 Jan 22						
92780A*	HP-ATS Automatic Test System	5 days	1000				Sep 25 Nov 27			
13294A	Dev. Terminal	5 days	500					Sep 11 Nov 13 Jan 8 Feb 26		
22940A	2100 Maint.	10 days	1000						Sep 11 Oct 9 Nov 6 Dec 4 Jan 22 Feb 26	
22941A	21MX/XE Maint.	5 days	500						Sep 11 Sep 25 Oct 16 Oct 30 Nov 27 Dec 4 Dec 11 Dec 18 Jan 8 Jan 15 Jan 29 Feb 5	
22942A	7900 Maint.	5 days	500						Oct 9 Nov 13 Jan 15 Feb 12	

*These courses carry prerequisites — refer to the data sheet for each course for more information.

U. S. TRAINING CENTER SCHEDULES, LOCATIONS, AND RATES (Continued)

Course Number	Title		CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Cupertino)	Boise Division (Boise)
	Length	Price							
22945A	7905 Maint.							Sep 25 Oct 2 Oct 23 Oct 30 Nov 27 Dec 18 Jan 8 Feb 5 Feb 12	
	5 days	500							
91302A	2645 Maint.							Sep 4	
	3 days	300							
22943A	7970B Maint.								Sep 25
	5 days	600							
22944A	7970E Maint.								Sep 18
	5 days	600							

*These courses carry prerequisites — refer to the data sheet for each course for more information.

U.S. TRAINING CENTER ADDRESSES

Cupertino

CUSTOMER TRAINING CENTER
19310 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9383

DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, CA 95014
(408) 257-7000

DATA TERMINALS DIVISION
19400 Homestead Road
Cupertino, CA 95014
(408) 257-7000

CUSTOMER SERVICE DIVISION
19310 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9383

Fullerton

CUSTOMER TRAINING CENTER
1430 E. Orangethorpe Avenue
Fullerton, CA 92631
(714) 870-1000

Rockville

CUSTOMER TRAINING CENTER
4 Choke Cherry Road
Rockville, MD 20850
(301) 948-6370

Boise

BOISE DIVISION
11311 Chinden Boulevard
Boise, Idaho 83702
(208) 377-3000

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title / Length	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
22965B	RTE-II/III 10 days (Course includes RTE-II/III operating system, batch spool monitor and file manager.)	Sep 25 Oct 23	Oct 9 Dec 4	Oct 23	Sep 4	Oct 9 (M)	Oct 9 Nov 27		Nov 6	Oct 9	Oct 2
22985A	RTE-M 5 days	Sep 18									
22977A*	IMAGE 5 days	Sep 4		Nov 6	Oct 23				Sep 25	Oct 23	
22952B*	1000 ASMB 5 days	Sep 11 Oct 23	Sep 11 Nov 6	Oct 16	Oct 16	Sep 11 (M)	Sep 11 Dec 11		Sep 18		
22987A*	DS/1000 User's Course 5 days	Oct 9									
22961B*	DS/1000 Theory of Op. 4 days										
22962B*	DS/1000 to HP 3000 Theory of Op. 1 day										
22990A*	RTE Driver Writing 3 days										
22980B*	HP-IB Minicomputer Environment 4 days		Oct 16 Dec 11								
22983A*	21MX-E Microprogramming 5 days										

*These courses carry prerequisites — refer to the data sheet for each course for more information.

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS (Continued)

Course Number	Title / Length	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
92780A*	HP-ATS Automatic Test System / 5 days										
13294A	Dev. Terminal / 5 days										
22940A	2100 Maint. / 10 days										
22941A	21MX/XE Maint. / 5 days							Oct 2 Feb 26			
22942A	7900 Maint. / 5 days							Sep 18 Dec 4 Mar 5			
22945A	7905/06 Maint. / 5 days							Oct 9 Dec 11 Feb 19 Apr 2			
22984A	7920 Maint. / 5 days							Nov 6 Apr 9			
91302A	2645 Maint. / 3 days							Oct 2 Jan 22			
22943A	7970B/E Maint. / 5 days							Sep 25 Jan 15 Mar 26			
40270A	Intro to HP Computers / 5 days	Sep 25							Oct 9		
22965B-H01	FORTRAN IV / 5 days	Oct 16		Oct 2							

These courses carry prerequisites. Refer to the data sheet for each course for more information.

EUROPEAN TRAINING CENTER ADDRESSES

Boblingen

Kundenschulung
Herrenbergerstrasse 110
D-7030 Boblingen, Wurttemberg
Tel: (07031) 667-1
Telex: 07265739
Cable: HEPAG

Brussels

Avenue du Col Vert, 1
Groenkraaglaan
B-1170
Brussels, Belgium
Tel: (02) 672 22 40

Stockholm

Enighetsvagen 1-3, Fack
S-161 20 Bromma 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Madrid

Jerez No. 3
E-Madrid 16
Tel: (1) 458 26 00
Telex: 23515 hpe

Amsterdam

Van Heuven Goedhartlaan 121
Amstelveen - 1134
Netherlands
Tel: 02 672 22 40

Orsay

Quartier de Courtaboeuf
Boite Postale No. 6
F-91401-Orsay
France
Tel: (01) 907 7825

Grenoble

5, avenue Raymond-Chanas
38320 Eybens
Tel: (76) 25-81-41
Telex: 980124

Vienna

Handelskai 52
Postfach 7
A - 1205 Wien
Tel: (0222) 35 16 21-32
Telex: 75923
Cable: Hewpack Wien

Milan

Via Amerigo Vespucci, 2
20124 Milan
Tel: (2) 62 51
Cable: HEWPACKIT Milano
Telex: 32046

Winnersh

King Street Lane
Winnersh, Workingham
Berkshire RG11 5 AR
Tel: Workingham 784774
Cable: Hewpie London
Telex: 8471789

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name _____ Date _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

☐ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	$48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6111				_____
5951-6112	COMMUNICATOR 2000	_____	25.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6112				_____
5951-6113	COMMUNICATOR 3000	_____	48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6113				_____

☐ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
5951-6112	COMMUNICATOR 2000	_____	_____	$ 5.00	_____	
		_____	_____	5.00	_____	
		_____	_____	5.00	_____	
	TOTAL DOLLARS					_____
5951-6113	COMMUNICATOR 3000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
TOTAL ORDER DOLLAR AMOUNT						_____

☐ **SERVICE CONTRACT CUSTOMERS**

You will receive one copy of either COMMUNICATOR 1000, 2000, or 3000 as part of your contract. Indicate additional copies below and have your local office forward. Billing will be included in normal contract invoices.

Number of additional copies _____

FOR HP USE ONLY

CONTRACT KEY

5951-6111 Number of additional copies _____
5951-6112 Number of additional copies _____
5951-6113 Number of additional copies _____

Approved _____

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

The Computer Systems COMMUNICATORS are bi-monthly systems support publications availahle from Hewlett-Packard on an annual (6 issues) subscription.

The following instructions are for customers who do not have Software Service Contracts.

1. Complete name and address portion of order form.
2. For new direct subscriptions (see sample below):
   a. Indicate which COMMUNICATOR publication(s) you wish to receive.
   b. Enter number of copies per issue under Qty column.
   c. Extend dollars (quantity x list price) in Extended Dollars column.
   d. Enter discount dollars on line under Extended Dollars. (If quantity is greater than 1 you are entitled to a 40% discount.*)
   e. Enter Total Dollars (subtract discount dollars from Extended List Price dollars).

   **To qualify for discount all copies of publications must be mailed to same name and address and ordered at the same time.*

SAMPLE

☒ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	3	$48.00	$144.00	
	(if quantity is greater than 1 discount is 40%)			57.60	
	TOTAL DOLLARS for 5951-6111				$86.40

3. To order back issues (see sample below):
   a. Indicate which publication you are ordering.
   b. Indicate which issue number you want.
   c. Enter number of copies per issue.
   d. Extend dollars for each issue.
   e. Enter total dollars for back issues ordered.

   All orders for back issues of the COMMUNICATORS are cash only orders (U.S. dollars only) and are subject to availability.

SAMPLE

☒ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	XX	1	$10.00	$10.00	
		XX	2	10.00	20.00	
				10.00		
	TOTAL DOLLARS					$30.00

4. Domestic Customers: Mail the order form with your U.S. Company Purchase Order or check (payable to Hewlett-Packard Co.) to:

   HEWLETT-PACKARD COMPANY
   Computer Systems COMMUNICATOR
   P.O. Box 61809
   Sunnyvale, CA 94088
   U.S.A.

5. International Customers: Order by part number through your local Hewlett-Packard Sales Office.

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name _____ Date _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

☐ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	$48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6111				_____
5951-6112	COMMUNICATOR 2000	_____	25.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6112				_____
5951-6113	COMMUNICATOR 3000	_____	48.00	_____	
	(if quantity is greater than 1 discount is 40%)			_____	
	TOTAL DOLLARS for 5951-6113				_____

☐ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
5951-6112	COMMUNICATOR 2000	_____	_____	$ 5.00	_____	
		_____	_____	5.00	_____	
		_____	_____	5.00	_____	
	TOTAL DOLLARS					_____
5951-6113	COMMUNICATOR 3000	_____	_____	$10.00	_____	
		_____	_____	10.00	_____	
		_____	_____	10.00	_____	
	TOTAL DOLLARS					_____
TOTAL ORDER DOLLAR AMOUNT						_____

☐ **SERVICE CONTRACT CUSTOMERS**

You will receive one copy of either COMMUNICATOR 1000, 2000, or 3000 as part of your contract. Indicate additional copies below and have your local office forward. Billing will be included in normal contract invoices.

Number of additional copies _____

FOR HP USE ONLY

CONTRACT KEY

5951-6111 Number of additional copies _____
5951-6112 Number of additional copies _____
5951-6113 Number of additional copies _____

Approved _____

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

The Computer Systems COMMUNICATORS are bi-monthly systems support publications available from Hewlett-Packard on an annual (6 issues) subscription.

The following instructions are for customers who do not have Software Service Contracts.

1. Complete name and address portion of order form.
2. For new direct subscriptions (see sample below):
   a. Indicate which COMMUNICATOR publication(s) you wish to receive.
   b. Enter number of copies per issue under Qty column.
   c. Extend dollars (quantity x list price) in Extended Dollars column.
   d. Enter discount dollars on line under Extended Dollars. (If quantity is greater than 1 you are entitled to a 40% discount.*)
   e. Enter Total Dollars (subtract discount dollars from Extended List Price dollars).

   **To qualify for discount all copies of publications must be mailed to same name and address and ordered at the same time.*

SAMPLE

☒ **DIRECT SUBSCRIPTION**

Part No.	Description	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	3	$48.00	$144.00	
	(if quantity is greater than 1 discount is 40%)			57.60	
	TOTAL DOLLARS for 5951-6111				$86.40

3. To order back issues (see sample below):
   a. Indicate which publication you are ordering.
   b. Indicate which issue number you want.
   c. Enter number of copies per issue.
   d. Extend dollars for each issue.
   e. Enter total dollars for back issues ordered.

   All orders for back issues of the COMMUNICATORS are cash only orders (U.S. dollars only) and are subject to availability.

SAMPLE

☒ **BACK ISSUE ORDER FORM (cash only in U.S. dollars)**
(subject to availability)

Part No.	Description	Issue No.	Qty	List Price	Extended Dollars	Total Dollars
5951-6111	COMMUNICATOR 1000	XX	1	$10.00	$10.00	
		XX	2	10.00	20.00	
				10.00		
	TOTAL DOLLARS					$30.00

4. Domestic Customers: Mail the order form with your U.S. Company Purchase Order or check (payable to Hewlett-Packard Co.) to:

   HEWLETT-PACKARD COMPANY
   Computer Systems COMMUNICATOR
   P.O. Box 61809
   Sunnyvale, CA 94088
   U.S.A.

5. International Customers: Order by part number through your local Hewlett-Packard Sales Office.

Please photocopy this order form if you do not want to cut the page off. You will automatically receive a new order form with your order.

HEWLETT hp PACKARD

CONTRIBUTED SOFTWARE
Direct Mail Order Form

NOTE: No direct mail order can be shipped outside the United States.

Please Print:

Name _____ **Title** _____

Company _____

Street _____

City _____ **State** _____ **Zip Code** _____

Country _____

Item No.	Part No.	Qty.	Description	List Price Each		Extended Total	

Sub-total		
Your State & Local Sales Taxes*		
Handling Charge	1	50
TOTAL		

*Tax is verified by computer according to your ZIP CODE. If no sales tax is added, your state exemption number must be provided: # _____.
If not, your order may have to be returned.

Domestic Customers: Cash required on all orders less than $50.00. Mail the order form with your check or money order (payable to Hewlett-Packard Co.) or your U.S. Company Purchase Order to:

HEWLETT-PACKARD COMPANY
Contributed Software
P.O. Box 61809
Sunnyvale, CA 94088

International Customers: Order through your local Hewlett-Packard Sales office. No direct mail order can be shipped outside the United States.

All prices domestic U.S.A. only. Prices are subject to change without notice.

ORDERING INFORMATION

Programs are available individually in source language on either paper tape, magnetic tape, or cassettes as indicated in the abstracts.

To order a particular program, it is necessary to specify the program identification number, together with an option number which indicates the type of product required. The program identification number with the option number composes the ordering number.

For example:

22113A-K01

The different options are.

K01 — Source paper tape and documentation
K21 — Magnetic tapes and documentation

NOTE

Specify 800 BPI or 1600 BPI Magnetic tape.

B01 — Binary tape and documentation
D00 — Documentation
L00 — Listing

Not all options are available for all programs.

Ten-digit numbers do not require additional option numbers such as K01, K21, etc. The 10-digit number automatically indicates the option or media ordered.

For example:

22681-18901 — The digits 189 indicate source paper tape plus documentation.
22681-10901 — The digits 109 indicate source magnetic tape plus documentation (800 BPI magnetic tape)
22681-11901 — The digits 119 indicate source magnetic tape plus documentation (1600 BPI magnetic tape)
22681-13301 — The digits 133 indicate source cassettes plus documentation

Only those options listed in each abstract are available.

Refer to the Price List for prices and correct order numbers.

Hewlett-Packard offers no warranty, expressed or implied and assumes no responsibility in connection with the program material listed.

HEWLETT-PACKARD
LOCUS CONTRIBUTED SOFTWARE CATALOG
DIRECT MAIL ORDER FORM

Please Print:

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

Part Number	Description	Qty.	List Price Each	Extended Total
22000-90099	Locus Contributed Software Catalog		$15.00	
*If no sales tax is added, your state exemption number must be provided: #_____ If not, your order may have to be returned.			Your State & Local Sales Taxes*	
			Handling Charge	1.50
			TOTAL	

Domestic Customers: Mail the order form with your check or money order (payable to Hewlett-Packard Co.) to:

HEWLETT-PACKARD COMPANY
LOCUS CATALOG
P.O. Box 61809
Sunnyvale, CA 94088

International Customers: Order by part number through your local Hewlett-Packard Sales Office.

NOTE: No direct mail order can be shipped outside the United States. All prices domestic U.S.A. only. Prices are subject to change without notice.

NOT TO BE USED FOR ORDERING COMMUNICATOR SUBSCRIPTIONS

HEWLETT hp PACKARD

CORPORATE PARTS CENTER

Direct Mail
Parts and Supplies Order Form

SHIP TO:

NAME _____

COMPANY _____ CUSTOMER REFERENCE # _____

STREET _____ TAXABLE*? _____

CITY _____ STATE _____ ZIP CODE _____

Item No.	Check Digit	Part No.	Qty.	Description	List Price Each		Extended Total	

Special Instructions

Sub-total		
Your State & Local Sales Taxes*		
Handling Charge	1	50
TOTAL		

*Tax is verified by computer according to your ZIP CODE. If no sales tax is added, your state exemption number must be provided: # _____. If not, your order may have to be returned.

Check or Money Order, made payable to Hewlett-Packard Company, must accompany order.

When completed, please mail this form with payment to:

HEWLETT-PACKARD COMPANY
Mail Order Department
P.O. Drawer #20
Mountain View, CA 94043

Phone: (415) 968-9200

Most orders are shipped within 24 hours of receipt. Shipments to California, Oregon and Washington will be made via UPS. Other shipments will be sent Air Parcel Post, with the exception that shipments over 25 pounds will be made via truck. No Direct Mail Order can be shipped outside the U.S.

Although every effort is made to ensure 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. 9/78

Part No. 5951-6111