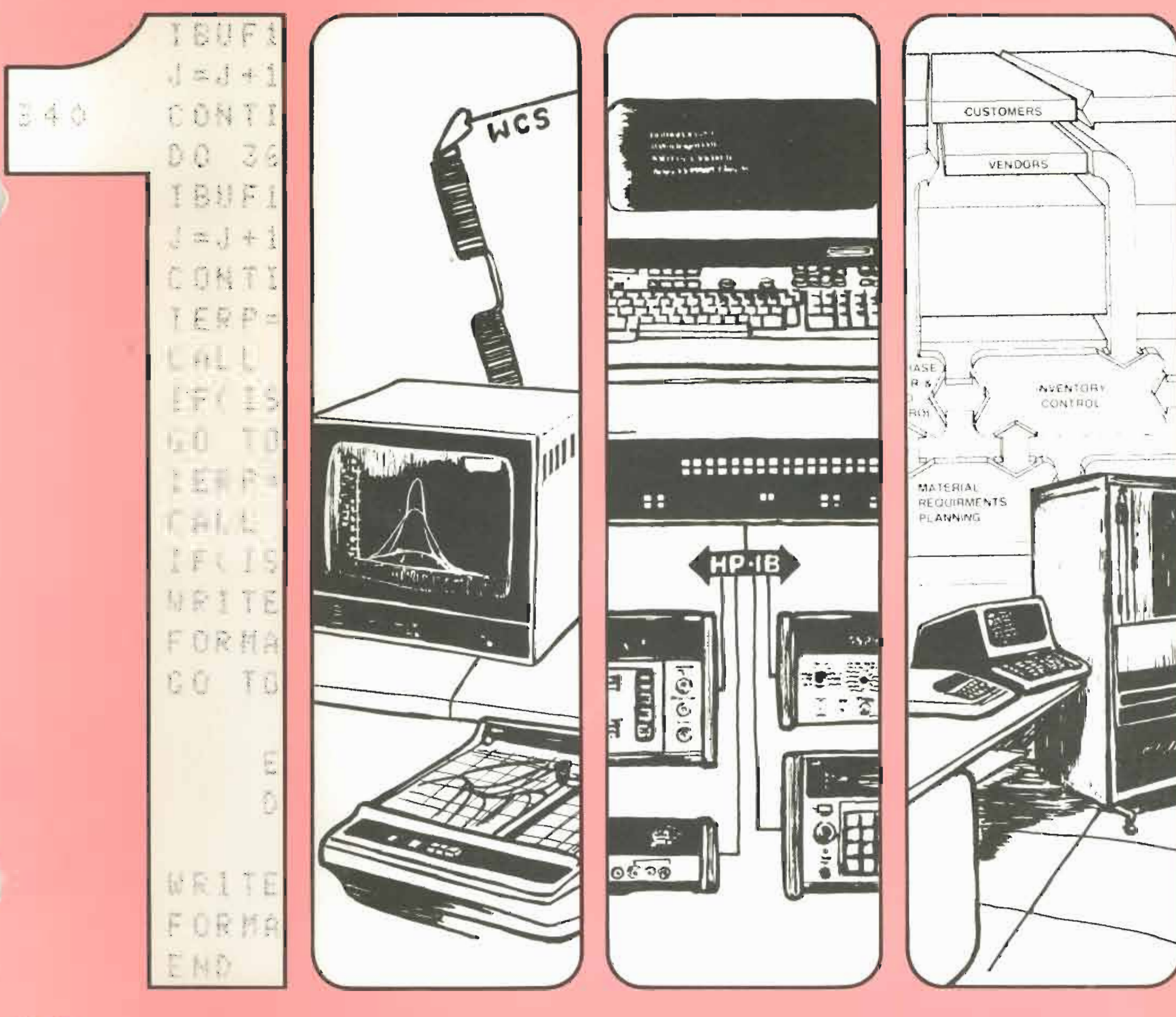
HEWLETT PACKARD

Computer Systems

COMMUNICATOR



1978

Volume II Issue No. 6

EDITOR'S DESK

If the last issue of the COMMUNICATOR was nicknamed DRIVER HINTS/1000 then perhaps this issue should be called EXTENDED MEMORY ARRAYS. Both Martha Robrahn and Larry Smith wrote articles on sharing EMA between programs for this issue, a feature which I would like to point out is NOT an HP supported feature. Van Diehl was kind enough to let us republish his article describing normal EMA for those of you who have not had exposure to EMA before. (Van's article was previously published in HP internal literature.)

ANNOUNCING CALCULATOR WINNERS!

Martha Robrahn of Hewlett-Packard in the Neely Los Angles office is this month's HP-32E winner for HP employees outside of Data Systems Division. Martha's article, "Sharing Extended Memory Arrays in RTE-IV", was judged to be best on the areas of clarity, completeness of subject covered and interest to the largest segment of our readership. Millo Fenzi is this month's HP-32E winner for HP employees inside Data System Division. Millo and Paul Streit, his co-author, wrote an excellent article titled "Data Capture in Manufacturing". Millo and Paul had no competition this month. Alas, once again we had no customer calculator winners. Unfortunately, we have no articles in the OEM Corner this month either.

CUMULATIVE COMMUNICATOR/1000 INDEX

This is the last issue of Volume 2. The next issue of the COMMUNICATOR will be Volume 3 issue 1. Thanks to your support there have been 22 issues of the COMMUNICATOR since it began publication in June of 1975. To commemorate this, a cumulative index of all past issues has been included in the BIT BUCKET of this issue.

DEADLINES

The tentative deadlines for submitting articles for Volume 3 are:

Issue 1	January 12, 1979
Issue 2	March 16, 1979
Issue 3	May 11, 1979
Issue 4	July 13, 1979
Issue 5	September 7, 1979
Issue 6	November 2, 1979

We hope that more readers will submit articles in Volume 3, either in hopes of winning a calculator, as a contribution to the OEM corner, as hints for other users in the Bit Bucket ar as letters for User's Queue or for Software Samantha.

CONTENTS

WIN AN HP-32E CALCULATOR!

Since its beginning in 1975, the Communicator has changed format several times. During this period, the primary source of technical articles has been employees of HP Data Systems Division. In order to increase the diversity of topics and number of articles we are soliciting articles from customers and other HP divisions. To make it worth your time, three free HP-32E hand-held calculators will be awarded per issue (one to a customer, one to an HP employee of HP Data Systems Division and one to an HP employee not from Data Systems Division) to the authors of the best feature-length articles which fall in one of the following categories:

Operating Systems
Instrumentation
Computation
Operations Management

The employees of the Technical Marketing department of Data Systems Division are not eligible for the calculator prize: all other HP employees are eligible. Customers and HP employees will not compete against each other, since HP employees have access to more information. Likewise, employees of Data Systems Division will not compete against employees not from Data Systems Division. A prize will be awarded even if there is only a single entry.

A feature-length article must meet the following criteria:

1. The topic must be of general interest to our readers and fall into one of the four categories above.

2. It must cover at least two pages in the 1000 Communicator, exclusive of listings and illustrations. At the current print size, this is approximately 1650 words.

The eligibility rules for receiving a calculator are:

1. No individual will be awarded more than one calculator per calender year.

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

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

4. Employees of Technical Marketing in HP Data Systems Division are not eligible.

The winning article will be the best article submitted based on the areas of clarity, completeness of subject coverage and interest to the largest segment of our readership. All entries will be judged by a team of at least three people in Technical Marketing.

All winners will be announced in the HP 1000 Communicator in the issue in which their articles appear. It is greatly appreciated if the text of the article and any listings are submitted in machine-readable form, i.e. a file on a magnetic tape, mini-cartridge or paper tape.

Address all communications to:

Editor HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

THE OEM CORNER OF THE HP-1000 COMMUNICATOR

Issue 3 marked the start of a major new section of the HP 1000 Communicator — the 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 a part of a system package which the OEM delivers as a "turnkey" package or a standalone 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.

In issue 3, we had "A Modern Language for On-Line Systems" by ***David C. Hamilton*** of Theta Computer Systems in Van Nuys, California.

In issue 4 there were no articles.

In issue 5, we have "Software for the 2645 Terminal" by ***P. Alex Swartz*** of Computer Systems Consultants of Tucson, Arizona.

To qualify for inclusion in OEM Corner, an article should be of general interest to our readers and have educational value. That is, it should describe a technique or method of doing something. The article should contain numberous examples and be application-oriented rather than theoretical. We encourage the OEM to describe as many of the features of his product as he wishes but, in all cases, we are looking for general inter-educational value. A reprint of a press release or a marketing brochure is not sufficiently technical to qualify.

We encourage the OEM to place, at the very end of the article, up to 150 words of purely commercial information. This may include prices of the product and ordering information.

It is expected that OEM software products complement the HP product line or present a more complete solution to a problem. HP, in contrast, sells tools of a general nature. Therefore, some explanation of this sort is permissable in the OEM's article. The article should present a technique or innovative idea of general interest to HP customers.

The readership of the Communicator is assumed to cover the full range from the neophyte to the expert. Therefore, the author may address any level of expertise he chooses. However, the clarity of presentation is always an important consideration, regardless of the assumed background of the reader.

The article should be a minimum of 4 typed, double-spaced pages. Only in unusual cases should an article be more than 10 type written pages.

All articles are subject to editorship and minor revisions. In general the author will be contacted if there is any question of changing the information content. Articles requiring major revision will be returned with an explanatory note. We hope not to return any articles and would like to work with all authors to avercome any objections. However, HP reserves the right to reject any articles judged not to be of general interest to HP customers.

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.

Address all Communications to:

Editor HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

NEW CONTRIBUTED PROGRAMS

Elisabeth Caloyannis/HP Data Systems Division

This article serves as an update for 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.

22683-10906 SPL/2100 — SPL Compiler

System Programming Language (SPL) is a high-level language for writing programs such as compilers, device drivers, and operating systems. SPL/2100 is a version designed for use with the HP 1000 series computer.

The SPL/2100 compiler can be used with an RTE-II, III or IV system. A program size of 15 pages should be specified when the compiler is loaded.

The compiler translates programs written in SPL/2100 to HP Assembly Language. The SPL program is input from the LS area and the resulting assembly language program is output back to the LS area. The compiler then schedules the HP Assember to process the assembly language program and generate the object code.

Since the RTE-IV assembler does not accept source programs from the LS area, the user must first save the LS area as a disc file and then schedule the assembler.

22683-10906	800 bpi MT	$70.00
22683-11906	1600 bpi MT	$70.00

22683-13307 TIME — HP-IB time program

TIME is a user program written in FORTRAN which accesses the 59309A HP-IB clock to set the time in an RTE-M (II and III), RTE-II, III or IV system. The primary purpose of TIME is to automatically set the system time in the WELCOM file during RTE boot-up. Simply enter:

```
:RU,TIME,1,1978,50
```

where 1 is the input LU, 1978 is the year, 50 is the 59309A LU and you wish to set the system time (CS). Actually, four commands can be used when the program is run interactively. CS accesses the clock, obtains the time and sets the system time. SC obtains the system time and then accesses the 59309A and updates it. OS allows an operator at a user terminal to update the 59309A from the program (without touching the clock) and simultaneously updates the system time. OC is similar to OS except that only the 59309A is updated.

```
:RU,TIME,input,year,59309A LU, command
```

If command=0 and the 59309A LU is non-zero, the command defaults to CS and terminates. If the 59309A LU is 0 the program is interactive.

CS — 59309A sets the system time
SC — System sets the 59309A clock
OS — Operator sets the 59309A clock which sets the system time
OC — Operator sets the 59309A clock

22683-13307	mini-cartridge	$60.00

22683-13308	TODAY — Date formatting program		

The TODAY subroutine translates system time into a date/time message in a 14 word buffer in the following format:

FRI 26 MAR 1976 18:24:30.09

where today's date is Friday, March 26, 1976 and the time is 6:24 PM and 30.09 seconds. Note that if the first two words and the last three words of the buffer are stripped off, the date would be displayed as:

26 MAR 1976 18:24

22683-13308 mini-cartridge $40.00

22683-13309 STRNG — conversion subroutine

Mixed groups of integer and real values, prefixed by one or two ASCII characters will be converted to a prefix, a real value, an integer value and an error indicator. The package contains three demostration programs using STRNG, one of which illustrates a pseudo-namelist capability.

22683-13309 mini-cartridge $50.00

22683-13310 POWER — outspool program

POWER will add versatility to your outspool applications. POWER offers forward and backward positioning, line and page counts, multiple copies, spool or standard format, and run cost based on CPU time when the standard JOB headers are present in the outspool file.

POWER has several features not available in GASP. Spooling can be shut down and POWER can handle the printing. A file may be restarted from any page or line in the event of printer malfunction. The file remains intact after printing until the operator issues the "KS" command.

POWER uses the "@" as a prompt character and will list all available functions if ?? is entered.

POWER locks the line printer and will respond to the break flag while printing. POWER requires a 7 page partition.

The source file is well commented so that on-site modifications can be made.

22683-13310 mini-cartridge $35.00

22683-13311 LISTB — list a FMP file

Program LISTB is a program to list a FMP file of any type to an output device in binary-mixed format or ASCII format. The program is especially useful for listing files with record lengths of greater than 128 words, and will handle files with record lengths up to 1024 words. The list format is similar to the file manager :LI command format, but requires 130 columns on a line printer. You may list the entire file, or only part of the file. LISTB requires an 8 page partition.

22683-13311 mini-cartridge $35.00

22683-13312 FDUMP -- dump a FMP file

FDUMP is a program that will dump FMP files of type 10 or less to the list device in a format much like that used in the IBM DITTO utility. The record is listed in 128 character blocks, giving octal representation and column alignment numbers below the ASCII values. The maximum record length is 1024 words. Integer values are not printed, however their octal equivalents are. Input is from file or logical unit. By specifying the starting record number you can position the file to a predetermined record. FDUMP will respond to the break flag at any time. FDUMP needs a 9 page partition.

22683-13312	mini-cartridge	$35.00

22683-13313 EBC2A — EBCDIC translation

Program EBC2A converts variable-length EBCDIC coded records from an input device or disc file to ASCII code and stores the translated records to an output device or disc file. Maximum allowable record length is 1024 words, but can be modified as required. Options are set through the RUN parameters. The options allow you to begin translation at a specified record and translate a given number of records. Defaults are to do input and output to a disc file and translate the entire file.

If the input or output is from the disc, the program will interactively prompt for the input and/or output file name. If the output file does not exist, EBC2A will create it.

The program requires 5 pages. If disc files are used, then spooling must be enabled for the system.

22682-13313	mini-cartridge	$35.00

22683-13314 A2EBC — ASCII to EBCDIC

Program A2EBC converts variable-length ASCII coded records from an input device or disc file to EBCDIC code and stores the translated records to an output device or disc file. Maximum allowable record length is 1024 words, but can be modified as required. Options are set through the RUN parameters. The options allow you to begin translation at a specified record and translate a given number of records. Defaults are to input and output to a disc file and translate the entire file.

If the input and/or output is from the disc, the program will interactively prompt for the input or output file names. If the output file does not exist, A2EBC will create it.

The program requires 5 pages. If disc files are used, then spooling must be enabled for the system.

22683-13314	mini-cartridge	$35.00

LETTERS

"Dear Editor (EDITR?),

Regarding "Operating Systems" miscellany on page 15 of Volume 2 issue 4: Using "PROGX" example it (the program) can become a six character type 6 file using

```
:SP,PROGX
:RN,PROGX,PROGX1
```

If the command :RU,PROGX1 is given, the program in the type 6 file is not executed if the original is still in the 'LOADR' because the 'RUN' command looks at the program list for first five characters before looking on LU2 and LU3 for type 6 files. In the case where the original is no longer in the program list, the type 6 file will be executed, but under the five character name.

I'm pleased with the effort of COMMUNICATOR to allow HP users to know each other, but I think a directory of HP 1000 users published in the COMMUNICATOR would help even more.

Sincerely,

David Welborn
Micro Craft, Inc.
Tullahoma, Tennessee 37389

P.S. How about publishing a Julian calendar in the COMMUNICATOR?"

Thank you for your comments. My name is indeed Editor although I am not too particular. You are correct that the FMGR will not run a program from a type 6 file if there is an ID segment for that program. If the case above is changed to:

```
:SP,PROGV
:RN,PRGV,PROGX1
:OF,PROGX,8
```

there will be no problem.

We are glad you find the COMMUNICATOR useful, but it is against HP policy to release a list of our customers.

Also, you will probably be glad to note that the Julian calendar in the BIT BUCKET is entirely due to your suggestion.

Thanks for writing

RECLAIMING CLASS NUMBERS

Dave R. Fullerton/HP Neely Santa Clara

The first time use of class I/O by the inexperienced user can be difficult. First, all those parameters and bits to set. Second, when the program aborts or the `CALL EXEC 21` is not quite perfect, the class number disappears, never to be seen again until re-boot. This problem can also plague the experienced programmer during program developement.

So, how to avoid the headaches and get the benefits? To simply avoid using class I/O is unacceptable since it does many unique things in RTE. (Some examples are "Using Class I/O in a Sort Application" in the HP 1000 COMMUNICATOR Volume 1 issue 16 and "Multiterminal I/O" in the HP 1000 COMMUNICATOR Volume 2 issue 2.) To help with the successful application of class I/O, two subroutines CGET and RELES are provided as examples, along with a program CLEAN to keep class numbers from dissappearing.

THE PROBLEMS

1. Allocating a class number.
2. De-allocating a class number.
3. A program aborts leaving garbage in System Available Memory (SAM) and the class number allocated.

THE SOLUTIONS

1. Subroutine **CGET** will request a class number from RTE and return it to the calling program.
2. Subroutine **RELES** will release all unclaimed buffers in SAM that are linked to the class number and then return the class number to RTE.
3. Program **CLEAN** will clear SAM and release any class number that was obtained from the routine CGET.

SOME BACKGROUND

When a request is made for a class number, an entry is made in a table in RTE. This entry stays there until specifically released by some program. Note that any program can release a class number — not just the program that requested it. So, while it is a definate feature that a program can get some class numbers, pass them to other programs and then go dormant, it also means that RTE cannot release resources when programs accidentally go away without releasing their class numbers.

This is typically a problem only during program development when "bugs" cause abnormal termination. If development is being done on a stand alone system even this is no cause for concern since it is very easy to re-boot and recover the system resources. However, if development is done on a production system, then shutting everything down to re-boot becomes a problem.

To avoid these situations it would be nice to store the class number somewhere, so that it could be released if the program fails to do it. This is what the subroutine CGET does. Besides allocating a class number and returning it to the calling program, it also logs the class number, the program name, and the time of day to a disk file named *CL.NO. The calling sequence is:

```
CALL CGET(ICLS,N1,N2,N3,LU)
```

where ICLS contains the class number on return, N1-N3 contain the name of the program, and LU is the lu number for error messages.

When the program is finished with the class number, a call to RELES will return the number to RTE and remove the log entry from the disk file. To call RELES:

```
CALL RELES(ICLS,LU)
```

where ICLS contains the class number, and LU is the lu for error messages.

To check the contents of the log file and/or remove an entry, use the program CLEAN. CLEAN will prompt you for the class number to de-allocate, then it will remove all buffers, return the number to RTE and, finally it will delete the entry from the log file.

THE FUTURE

While these routines work well for keeping track of class number allocaton, they require that the user manually clear the class number with CLEAN if there is a failure. It would be nice if CLEAN could be modified so that is would clear the number automatically. In fact this can be extremely important in certain real-time applications. The challenge to all readers is this: is there a way to have a program in the time schedule list that periodically checks the class number log file to see if anybody went away without clearing SAM? Remember, the program could have gone dormant and another program or programs are using the class number, so the problem is not an easy one. I look forward to reading possible solutions to this problem in future issues of the COMMUNICATOR.

EDITOR'S NOTE: I feel that this problem is non-deterministic and look forward to reading a rigerous proof of this in some future issue of the COMMUNICATOR.

```
PAGE 0001  FTN.   5:48 PM  WED., 29  NOV., 1978

FTN4,L
      SUBROUTINE CGET(ICLS,N1,N2,N3,LU)
      DIMENSION NAME(3),ISZ(2),IB(8),IBUF(8),IDCB(144),IT(5)
      DATA NAME/2H*C,2HL.,2HNO/,ISZ/1,8/,ISC/2HCL/
C
C     THIS SUBROUTINE WILL ALLOCATE A CLASS NUMBER AND LOG
C     THAT TRANSACTION TO A DISC FILE
C     THIS WAY IF THE CALLING PROPGRAM ABORTS THE NUMBER
C     MAY BE RELEASED BY SOME OTHER PROGRAM
C
C     TO CALL THIS ROUTINE IN FORTRAN
C      CALL CGET(CLASS-NUMBER VARIABLE,ID1,ID2,ID3,LU-NUMBER FOR ERROR MSG
C     WHERE ID1,ID2,ID3 ARE SOME PROGRAM ID FOR THE LOG FILE
C     SUCH AS THE PROGRAM NAME
C
C     THE DATA IS STORED IN A FILE *CL.NO
C     DATA IS WRITTEN TO THE DISC IN A TYPE 2 FILE
C     EACH RECORD IS 8 WORDS LONG
C     WORD(1)=THE CLASS NUMBER
C     WORD(2)=THE PROGRAM NAME- 1
C     WORD(3)=THE PROGRAM NAME- 2
C     WORD(4)=THE PROGRAM NAME- 3
C     WORD(5)=THE TIME OF THE TRANSACTION (DAY)
C     WORD(6)=THE TIME OF THE TRANSACTION (HOUR)
C     WORD(7)=THE TIME OF THE TRANSACTION (MINUTES)
C     WORD(8)=THE TIME OF THE TRANSACTION (SECONDS)
C
C     * THE FIRST RECORD IS THE "DIRECTORY"
C     * IT CONTAINS HOW MANY ENTRIES ARE IN THE FILE
C
      ICLS=0
C
C     GET A CLASS # FROM THE SYSTEM
C
      CALL EXEC(20,0,I,1,IP1,IP2,ICLS)
C
C     SET THE DO-NOT DEALLOCATE BIT
C
      ICLS=ICLS+20000B
C
C     CLEAR THE CONTROL INFO IN SAM THAT WAS GENERATED BY THE ALLOCATE
C
      CALL EXEC(21,ICLS,I,1)
C
C     CLASS NUMBER IS O.K. - **HOWEVER** IF I WERE A GOOD PROGRAMMER
C     I WOULD HAVE PUT ERROR CHECKS IN ALL THE EXEC CALLS
C
C     LOG CLASS NUMBER TO DISC
C
C     DOES THE FILE EXIST YET?? - TRY A CREATE AND SEE
C
      CALL CREAT(IDCB,IER,NAME,ISZ,2,ISC,0)
C
C     SEE IF IT WAS CREATED
C
```

```
PAGE 0002  CGET    5:48 PM  WED., 29  NOV., 1978

           IF(IER.GE.0)  GO TO 222
      C
      C    SEE IF IT EXISTS
      C
           IF(IER.EQ.-2) GO TO 225
      C
      C    HERE IF SOME THING WRONG
      C
           IF(IER.LT.0) CALL ERROR(LU,IER,1)
      C
      C    HERE IF NEW FILE-SO OUTPUT HEADER RECORD
      C
       222 IBUF(1)=2
           CALL WRITF(IDCB,IER,IBUF,8,1)
           IF(IER.LT.0) CALL ERROR(LU,IER,2)
      C
      C    HERE TO ADD NEW CLASS # TO FILE
      C    FIRST GET AVAILABLE RECORD NUMBER
      C
       225 CALL OPEN(IDCB,IER,NAME,2,ISC)
           IF(IER.LT.0) CALL ERROR(LU,IER,3)
           CALL READF(IDCB,IER,IBUF,8,IP1,1)
           IF(IER.LT.0) CALL ERROR(LU,IER,4)
      C
      C    IBUF(1)=FIRST AVAILABLE RECORD
      C
      C    WRITE DATA TO FILE
      C
           IB(1)=ICLS
           IB(2)=N1
           IB(3)=N2
           IB(4)=N3
      C
      C    GET TIME
      C
           CALL EXEC(11,IT)
           IB(5)=IT(5)
           IB(6)=IT(4)
           IB(7)=IT(3)
           IB(8)=IT(2)
      C
      C    LOG TO DISC
      C
           CALL WRITF(IDCB,IER,IB,8,IBUF(1))
           IF(IER.LT.0) CALL ERROR(LU,IER,5)
      C
      C    UPDATE DIRECTORY
      C
           IBUF(1)=IBUF(1)+1
           CALL WRITF(IDCB,IER,IBUF,8,1)
           IF(IER.LT.0)      CALL ERROR(LU,IER,6)
           CALL CLOSE(IDCB)
           RETURN
           END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00414      COMMON = 00000
```

Figure 1. Subroutine CGET

```
PAGE 0001  FTN.   5:37 PM  WED., 29  NOV., 1978

FTN4,L
      SUBROUTINE RELES(ICLS,LU)
      DIMENSION IDCB(144),NAME(3),IBUF(8),IB(8)
      DATA NAME/2H*C,2HL.,2HNO/,ISC/2HCL/
C
C     THIS ROUTINE WILL DE-ALLOCATE A CLASS NUMBER AND REMOVE ITS
C     ENTRY FROM THE LOG FILE *CL.NO
C
C     TO CALL THIS ROUTINE IN FORTRAN
C       CALL RELES(CLASS-NUMBER,LU)
C
C     THIS ROUTINE WORKS WITH SUBROUTINE CGET
C
C
C     SET THE NO WAIT BIT
C
      ICLS=IOR(ICLS,100000B)
C
C     NOW - GO IN A LOOP AND DO CLASS GETS
C     THIS IS TO CLEAR OUT ANY "OLD" DATA LEFT IN SAM
C     BY THIS NUMBER.
C     THIS WAY WE CLEAN UP SAM
C     AND THEN WHEN THERE IS NO DATA
C     (WHICH WE CHECK BY BIT 15 OF THE "A" REGISTER)
C     THE NUMBER CAN BE RELEASED
C
 1    CALL EXEC(21,ICLS,I,1)
C
C     CHECK THE A REGISTER
C
      CALL ABREG(IA,IB)
C
C     CHECK BIT 15
C
      IF(IA.LT.0) GO TO 222
C
C     HERE IF BIT 15 0
C     SO THERE IS DATA IN SAM
C     GO BACK AND TRY AGAIN
C     UNTIL SAM IS CLEAR
C
      GO TO 1
C
C
C     HERE WHEN THERE ARE NO MORE BUFERS IN SAM FOR THIS
C     CLASS NUMBER
C
C     CLEAR THE DO NOT DE-ALLOCATE BIT AND NO WAIT BITS
C
 222  ICLS=IAND(ICLS,17777B)
C
C     RELEASE NUMBER
C
      CALL EXEC(21,ICLS,I,0)
C
```

```
PAGE 0002  RELES  5:37 PM  WED., 29  NOV., 1978
C      REMOVE ENTRY FROM THE DISC FILE
C
       ICLS=ICLS+20000B
       CALL OPEN(IDCB,IER,NAME,2,ISC)
       IF(IER.LT.0) CALL ERROR(LU,IER,10)
C
C      GET THE NUMBER OF ENTRIES FROM THE "DIRECTORY"
C
       CALL READF(IDCB,IER,IBUF,8,IP1,1)
C
C      IBUF(1)=THE NUMBER OF ENTRIES
C
       DO 15 I=2,IBUF(1)
       CALL READF(IDCB,IER,IB,8,IP1,I)
       IF(IER.LT.0) CALL ERROR(LU,IER,11)
       IF(ICLS.EQ.IB(1)) GO TO 300
C
C      NO MATCH SO GO TRY AGAIN
C
 15    CONTINUE
C
C      HERE IF NO MATCH IN FILE
C
       WRITE(LU,123) ICLS
 123   FORMAT(" ** NO MATCH FOR CLASS NUMBER ** ",O8)
       PAUSE 123
C
C      HERE IF MATCH IN FILE
C
C      NOW DELETE ENTRY FROM FILE
C       IF ITS THE LAST ENTRY IN THE FILE WE
C      ARE IN LUCK - ALL THAT NEEDS TO BE DONE
C      IS TO DECREMENT THE "DIRECTORY"
 300   IF(I.EQ.IBUF(1)) GO TO 900
C
C      NO SUCH LUCK
C      SO NOW WE HAVE TO MOVE UP ALL THE RECORDS BELOW
C      THE DELETED ENTRY
C
       DO 20 J=I,IBUF(1)-2
       CALL READF(IDCB,IER,IB,8,IP1,J+1)
       IF(IER.LT.0) CALL ERROR(LU,IER,12)
       CALL WRITF(IDCB,IER,IB,8,J)
       IF(IER.LT.0) CALL ERROR(LU,IER,13)
 20    CONTINUE
C
C      NOW UPDATE THE "DIRECTORY"
C
 900   IBUF(1)=IBUF(1)-1
       CALL WRITF(IDCB,IER,IBUF,8,1)
       IF(IER.LT.0) CALL ERROR(LU,IER,14)
C
C      CLOSE FILE AND GO HOME
C
       CALL CLOSE(IDCB)
       RETURN
       END
  ** NO WARNINGS ** NO ERRORS **  PROGRAM = 00415     COMMON = 00001
```

Figure 2. Subroutine RELES

```
PAGE 0001  FTN.   5:52 PM  WED., 29  NOV., 1978

FTN4,L
      SUBROUTINE ERROR(LU,IER,IP)
      WRITE(LU,100) IER,IP
  100 FORMAT(" ** ERROR ",I3," *** AT PROGRAM LOCATION ",I2)
      PAUSE 3
      END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00046      COMMON = 00000
```

Figure 3. Subroutine ERROR

```
PAGE 0001  FTN.   5:54 PM  WED., 29  NOV., 1978

FTN4,L
      PROGRAM CLEAN
      DIMENSION IP(5),IDCB(144),NAME(3),IB(8),IBUF(8)
      DATA NAME/2H*C,2HL.,2HNO/,ISC/2HCL/
C
C     THIS PROGRAM WILL LIST THE TABLE OF CLASS NUMBERS
C     IN FILE *CL.NO
C     THIS PROGRAM WORKS IN CONJUNCTION WITH SUBROUTINES
C     CGET AND RELES
C
C     THREE COMMANDS ARE AVAILABLE:
C     LIST-TO LIST THE CONTENTS OF THE FILE
C     RELEASE-TO RELEASE A CLASS NUMBER IN THE FILE
C     STOP-TO STOP THE PROGRAM
C
      CALL RMPAR(IP)
      LU=IP(1)
      IF(LU.LE.0) LU=1
    1 WRITE(LU,100)
  100 FORMAT(" ENTER LI TO LIST, RE TO CLEAR A #, OR STOP _")
      READ(LU,110) J
  110 FORMAT(A2)
      IF(J.EQ.2HLI) GO TO 300
      IF(J.EQ.2HRE) GO TO 330
      IF(J.EQ.2HST) GO TO 999
      WRITE(LU,120)
  120 FORMAT(" ??WHAT??")
      GO TO 1
C
C     HERE TO LIST THE FILE
C
  300 CALL OPEN(IDCB,IER,NAME,2,ISC)
      IF(IER.LT.0) CALL ERROR(LU,IER,1)
C
C     GET THE NUMBER OF ENTRIES IN THE FILE FROM THE "DIRECTORY"
C
      CALL READF(IDCB,IER,IBUF,8,IP1,1)
C
C     IBUF(1)=THE POINTER TO THE FIRST AVAILABLE RECORD
C     DO IBUF(1)-1=THE NUMBER OF RECORDS IN THE FILE
C
      IF(IBUF(1).GT.2) GO TO 400
      WRITE(LU,130)
  130 FORMAT(" SORRY - THERE ARE NO ENTRIES IN THE LOG FILE")
      GO TO 1
```

```
C
C     THERE IS SOMETHING IN THE FILE IF HERE SO PRINT!
C
 400  WRITE(LU,150)
 150  FORMAT(///" #  CLASS #  PROGRAM NAME   TIME OF ENTRY DAY:HR:MM:SS"
     1)
      DO 15 I=2,IBUF(1)-1
      CALL READF(IDCB,IER,IB,8,IP1,I)
      IF(IER.LT.0) CALL ERROR(LU,IER,2)
      WRITE(LU,160) I,IB

PAGE 0002  CLEAN  5:54 PM  WED., 29  NOV., 1978

 160  FORMAT(     I2,2X,I6,5X,3A2,10X,I3,":",I2,":",I2,":",I2)
 15   CONTINUE
C
C     HERE WHEN DONE PRINTING
C
      GO TO 1
C
C     HERE TO RELEASE A CLASS NUMBER
C
 330  WRITE(LU,170)
 170  FORMAT(" WHICH NUMBER TO RELEASE ?? (PLEASE ENTER CLASS #)")
      READ(LU,*) J
      CALL RELES(J,LU)
      GO TO 1
 999  END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00496      COMMON = 00000
```

Figure 4. Program CLEAN

```
PAGE 0001  FTN.    5:45 PM  WED., 29  NOV., 1978

FTN4,L
      PROGRAM TEST
C
C     THIS PROGRAM CALLS A ROUTINE TO OBTAIN A CLASS #
C
      DIMENSION IP(5)
      DATA N1/2HTE/,N2/2HST/,N3/2H  /
      CALL RMPAR(IP)
      LU=IP(1)
      IF(LU.LE.0) LU=1
C
C     GET A CLASS #
C     ICLS HAS A VALID CLASS # ON RETURN
C     *** WITH THE DO-NOT DEALLOCATE BIT SET (BIT 13) ***
C     N1-N3 HAS THE PROGRAM ID OR NAME
C
      WRITE(LU,100)
  100 FORMAT("GETTING CLASS #")
      CALL CGET(ICLS,N1,N2,N3,LU)
      WRITE(LU,101)
  101 FORMAT("HAVE NUMBER")
      PAUSE 777
      CALL RELES(ICLS,LU)
      WRITE(LU,123)
  123 FORMAT(" CL ALL GONE")
C
      END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00093      COMMON = 00000
```

Figure 5. Program TEST

HP-IB

CONTROLLING THE 8660A/B/C WITH HP-IB

Neal Kuhn/HP Data Systems Division

This article presents a group of device subroutines to control the HP8660A/B/C through HP-IB. The HP8660A/B/C is a Synthesized Signal Generator. The A/B/C in this article refers to the A,B, and C versions. When equipped with the HP-IB option, most of the 8660A/B/C's functions, such as frequency, output level, and modulation can be controlled by a computing controller such as the HP 1000.

The HP8660A/B/C was one of the original HP-IB devices (created even before the IEEE Standard), and presents some requirements that prevent straightforward HP-IB control. The 8660A/B/C was retrofitted to allow HP-IB operation, and as a result, the 8660A/B/C expects all data strings to be sent to it in reverse order (least significant digit to most significant digit). Leading zeroes are also needed, and each function has a specific requirement for the number of allowable digits. For example, frequency is programmed in hertz with 10 significant digits. The frequency 57.34 Mhz is 0057340000 hz (to 10 digits). Reversing the string yields 0000437500. That is the string needed by the 8660A/B/C.

PROGRAMMABLE SIGNIFICANT DIGITS

Function	Number of Significant Digits
Frequency	10
Output Level	3
AM	2
FM Deviation	2

A generalized method of string reversal is not simple when using the HP 1000. The reason is that the HP 1000 stores floating point (and double precision) values in "logarithimic" format as an exponent and mantissa in base 2. Division by 10 is not just a shifting of the decimal place, but a full floating point operation.

The solution to 8660A/B/C control is the creation of a group of device subroutines. These subroutines would reverse the data, translate to a proper reference level, zero fill as appropriate, and send the data to the 8660A/B/C complete with the proper control characters.

The listing that follow contain five routines. Four are device subroutines to set frequency, amplitude modulation, FM deviation, and output level. The fifth routine is a utility program to run diagnostic tests with the 8660A/B/C. With this utility program, any of the above four parameters can be sent from a terminal.

LOADING AND RUNNING

All routines were written in FORTRAN IV. Either load the appropriate subroutines with a user written main program, or load all four subroutines along with the utility program. The utility program uses the new (*MESS) HP-IB library.

Each device subroutine requires two parameters. The first is the LU of the 8660A/B/C. The second parameter is the value for the respective function to be performed. Note that the RFF (frequency set) routine requires a double precision value. All other parameter values are real, and of course, the LU values are integers.

The utility program looks for and requires three parameters when run. The first parameter is the LU of the terminal that you are operating from. The program will obtain it if you leave the parameter blank. The second parameter is the LU of the HP-IB, and the third is the LU of the 8660. The calling sequence will look like:

```
:RU,T8660,LUTERM,LUBUS,LU8660
```

The utility program will then interactively prompt the user for commands. First it will ask for a function to perform, then for a value. The set of functions which the program recognizes is:

A	set AM modulation
D	set FM deviation
F	set frequency (double precision required)
L	set RF level
S	STOP

The device subroutines shown control the major functions for the HP8660A/B/C. There are other functions which can be programmed. For a complete discussion on the 8660A/B/C, refer to the operation and service manual for the 8660A/B/C (pn 08660-90046), and the HP-IB Users Guide for the HP 1000 (pn 59310-90064). Also, Application Note 164-2, "Calculator Control of the 8660A/B/C Synthesized Signal Generator" details most of the information needed to control the synthesizer on HP-IB.

```
PAGE 0001  FTN.   4:11 PM  THU., 30  NOV., 1978

FTN4,L
      PROGRAM T8660
C
C     THIS IS A UTILITY ROUTINE TO DRIVE THE 8660 SYNTHESIZED SIGNAL
C     GENERATOR.  THIS ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:
C
C         :---------------------------------------------:
C         : OPERATION CODE :         FUNCTION           :
C         :----------------:----------------------------:
C         :       A        :         SETS AM MODULATION :
C         :       D        :         SETS FM DEVIATION  :
C         :       F        :         SETS FREQUENCY     :
C         :       L        :         SETS LEVEL         :
C         :       S        :         STOP               :
C         :---------------------------------------------:
C
C     THE CALLING PARAMETERS FOR THIS ROUTINE ARE:
C
C     :RU,T8660,TERMINAL,BUSLU,8660LU
C
C     WHERE TERMINAL IS THE LU OF YOUR TERMINAL,
C           BUSLU IS THE LU OF THE HPIB BUS
C       AND 8660LU IS THE LU FOR THE 8660.
      INTEGER IP(5),TLU,BLU
      DOUBLE PRECISION DVAL
      CALL RMPAR(IP)
      TLU=IP
      IF(IP.EQ.0) TLU=1
      ILU=IP(3)
      BLU=IP(2)
      CALL RMOTE(BLU)
      WRITE(ILU,333)
 333  FORMAT("/1000(351C88$00%")
 66   WRITE(TLU,101)
 101  FORMAT("ENTER COMAND")
      READ(TLU,102) ICMD
 102  FORMAT(1A1)
      IF(ICMD.EQ.1HA) GO TO 11
      IF(ICMD.EQ.1HD) GO TO 22
      IF(ICMD.EQ.1HL) GO TO 33
      IF(ICMD.EQ.1HF) GO TO 44
      IF(ICMD.EQ.1HS) GO TO 99
      GO TO 88
 103  FORMAT("ENTER VALUE _")
 11   WRITE(TLU,103)
      READ(TLU,*) VAL
      CALL RFA(ILU,VAL)
      GO TO 66
 22   WRITE(TLU,103)
      READ(TLU,*) VAL
      CALL RFD(ILU,VAL)
      GO TO 66
 33   WRITE(TLU,103)
      READ(TLU,*) VAL
      CALL RFL(ILU,VAL)
```

```
PAGE 0002  T8660  4:11 PM  THU., 30  NOV., 1978

           GO TO 66
     44    WRITE(TLU,103)
           READ(TLU,*) DVAL
           CALL RFF(ILU,DVAL)
           GO TO 66
     88    WRITE(TLU,104)
     104   FORMAT("BAD COMMAND--TRY AGAIN")
           GO TO 66
     99    STOP
           END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00230      COMMON = 00000
```

```
PAGE 0001  FTN.   4:15 PM  THU., 30  NOV., 1978

     FTN4,L
           SUBROUTINE RFA(DLU,AMP), 8660 AM MODULATION SET NHK-5/78
     C
     C     THIS ROUTINE SETS UP AM MODULATION AND THE % MODULATION FOR THE 8660
     C     THE PROGRAM RECEIVES THE LU OF THE 8660 AND THE PERCENTAGE MODULATION
     C     THE PROGRAM REVERSES THE ORDER OF THE MODULATION DIGITS, AND SENDS
     C     THE PROPER CHARACTERS TO THE 8660 TO SET UP AM, AND THE PERCENTAGE
     C     REQUESTED.
           INTEGER IBUF1(2),IBUF2(2),DLU
           CALL CODE
           WRITE(IBUF1,101) AMP
     101   FORMAT(1I2)
           IL=IAND(IBUF1(1)/400B,377B)
           IF(IL.EQ.040B) IL=060B
           IH=IAND(IBUF1(1),377B)
           IF(IH.EQ.040B) IH=060B
     88    IBUF2(1)=IH*400B+IL
           WRITE(DLU,102)IBUF2(1)
     102   FORMAT("88$",1A2,"%")
           RETURN
           END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00093      COMMON = 00000
```

```
PAGE 0001  FTN.   4:15 PM  THU., 30  NOV., 1978

      FTN4,L
            SUBROUTINE RFD(DLU,DEV), 8660 FM DEVIATION  SET NHK-5/78
      C
      C     THIS ROUTINE SETS UP FM DEVIATION FOR THE 8660.  THE ROUTINE
      C     RECEIVES THE LU OF THE 8660, AND THE AMOUNT OF DEVIATION.  SINCE
      C     ONLY TWO DIGITS ARE SENT OUT TO THE 8660, THIS ROUTINE WILL
      C     DETERMINE THE SCALING OF THE VALUE (LESS THAN OR GREATER THAN
      C     10 KILOHERTZ).  THE ROUTINE THEN REVERSES THE DIGITS, AND SENDS
      C     CONTROL CHARACTERS TO SET THE MODE AND SOURCE, THE VALUE, AND THE
      C     PROPER TERMINATOR CHARACTER.
            INTEGER IBUF1(2),IBUF2(2),DLU
            IRNG=0
            IF(DEV.LT.10.0) GO TO 88
            CALL CODE
            WRITE(IBUF1,101) DEV
      101   FORMAT(1I2)
      44    IL=IAND(IBUF1(1)/400B,377B)
            IF(IL.EQ.040B) IL=060B
            IH=IAND(IBUF1(1),377B)
            IF(IH.EQ.040B) IH=060B
            IBUF2(1)=IH*400B+IL
            IRNG=2+(IRNG*2)
            WRITE(DLU,102)IRNG,IBUF2(1)
      102   FORMAT("8",I1,"$",1A2,"%")
            RETURN
      88    IRNG=1
            A=DEV*10
            CALL CODE
            WRITE(IBUF1,101) A
            GO TO 44
            END

   **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00138     COMMON = 00000
```

```
PAGE 0001  FTN.   4:15 PM  THU., 30  NOV., 1978

      FTN4,L
            SUBROUTINE RFF(DLU,FRQ), 8660 FREQUENCY SET NHK-5/78
      C
            INTEGER IBUF1(5),IBUF2(5),DLU
      C     THIS ROUTINE SETS UP THE FREQUENCY FOR THE 8660.  THE FREQUENCY
      C     IS SENT TO THIS ROUTIE AS A DOUBLE PRECISION VALUE SINCE TEN DIGITS
      C     ARE REQUIRED.  THE ROUTINE RECEIVES THE LU OF THE 8660 AND THE FREQ.
      C     IT REVERSES THE ORDER OF THE DIGITS, AND SENDS THE VALUE TO THE 8660
            DOUBLE PRECISION FRQ,A
            A=FRQ*1E6
            CALL CODE
            WRITE(IBUF1,101) A
      101   FORMAT(1I10)
            DO 88 I=1,5
            IL=IAND(IBUF1(I)/400B,377B)
            IF(IL.EQ.040B) IL=060B
            IH=IAND(IBUF1(I),377B)
            IF(IH.EQ.040B) IH=060B
      88    IBUF2(6-I)=IH*400B+IL
            WRITE(DLU,102)IBUF2
      102   FORMAT(5A2,"(")
            RETURN
            END

   **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00130     COMMON = 00000
```

```
PAGE 0001  FTN.   4:16 PM  THU., 30  NOV., 1978

      FTN4,L
            SUBROUTINE RFL(DLU,LVL), 8660 RF OUTPUT LEVEL NHK-5/78
      C
      C     THIS IS A ROUTINE TO SET OUTPUT LEVEL TO THE 8660. THE PROGRAM
      C     RECEIVES THE LU OF THE SIG GEN, AND THE LEVEL IN DBM.  THIS
      C     ROUTINE REFERENCES THE LEVEL TO 13 DBM, REVERSES THE ORDER
      C     OF THE DIGITS AND OUTPUTS THEM TO THE 8660 LU WITH THE PROPER
      C     CONTROL CHARACTER.
            INTEGER IBUF1(2),IBUF2(2),DLU
            REAL LVL
            A=ABS(13-LVL)
            CALL CODE
            WRITE(IBUF1,101) A
      101   FORMAT(1I4)
            DO 88 I=1,2
            IL=IAND(IBUF1(I)/400B,377B)
            IF(IL.EQ.040B) IL=060B
            IH=IAND(IBUF1(I),377B)
            IF(IH.EQ.040B) IH=060B
      88    IBUF2(3-I)=IH*400B+IL
            IBUF2(2)=IAND(IBUF2(2),177400B)
            IBUF2(2)=IBUF2(2)+103B
            WRITE(DLU,102) IBUF2
      102   FORMAT(2A2)
            RETURN
            END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00128      COMMON = 00000
```

EXTENDED MEMORY ARRAYS

Van Diehl/HP Data Systems Division

Extended memory area (EMA) is an area for arrays limited only by the size of the physical memory. Note that one or many arrays may reside in the EMA and that these arrays may be small or very large. An EMA can extend well beyond the maximum program addressable space. It occupies the available memory in the program's partition that extends beyond the program's logical address space. (Figure 1).

A section of the EMA, two pages or more, must be included within the program's logical address for the mapping of a window segment (MSEG) of EMA. When a program accesses an array element that is not in the program logic address space, a window around this element in EMA is mapped into MSEG, inside the program logic address space.

This mapping requires no disc swaps, therefore, it is very fast. STANDARD FORTRAN I/O AND ARRAY ACCESSES USING SUBSCRIPTS ARE HANDLED WITHOUT ANY SPECIAL ACTION by the user. In FORTRAN, EMA arrays are used just like any other array. Several sub-partitions can be defined on the area occupied by the mother partition. Thus, once the EMA use is finished, the memory is available for other uses. A segmented program may use EMA. This allows many separate operations to be performed on the same EMA, e.g., one segment reads the data, a second processes it and a third saves the results. (Figure 2.)

Extended memory areas are used for large amounts of data storage, acquisition and processing. Accessing data within EMA does not involve any disc access, therefore, it is quite fast. EMA's are useful for data acquisition from fast devices at real time rates. EMA's would also be very useful in data processing that requires a lot of data accessing from random locations (e.g., sorting). Scientific applications using large matrices, like inverting a matrix, can be performed with ease and speed.

The beauty of EMA is that you can write programs in FORTRAN, using large data arrays, without any special user coded data management functions.

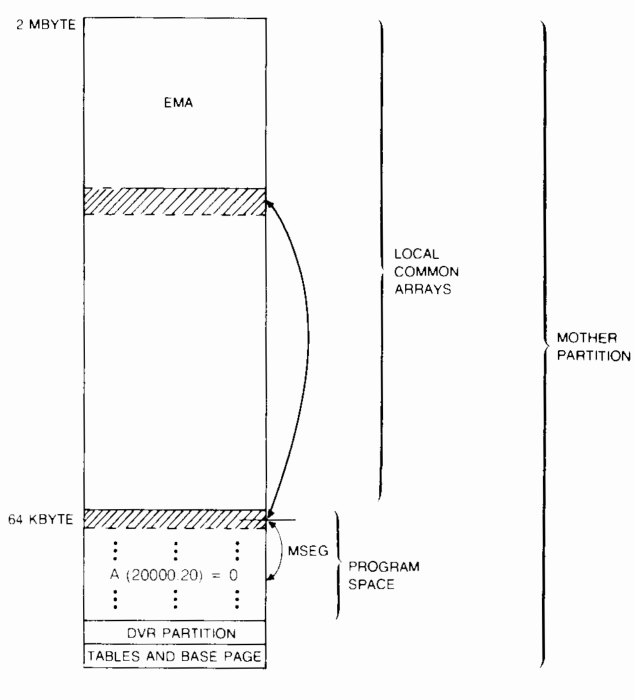


Figure 1

It should be noted that programs using data in EMA will not always run faster than programs using data on disc. It is possible via special user-coded data management functions to have programs with data stored on disc, running faster than FORTRAN programs using EMA.

However, the comparison here is not straight apples-for-apples, because all EMA programs can be made to run faster than disc programs if the user does his own mapping in assembler language.

This word of caution is added here because a user may convert an existing program, using some specially coded virtual data management scheme and expect that the program will, in all cases, run faster using data in memory.

The features of EMA can thus be summarized as:

- Easy FORTRAN coding of large array manipulation programs.
- Fast retrieval of random access data
- Virtual data in memory is faster than virtual data on disc
- Fast retrieval of sequential data with user custom mapping

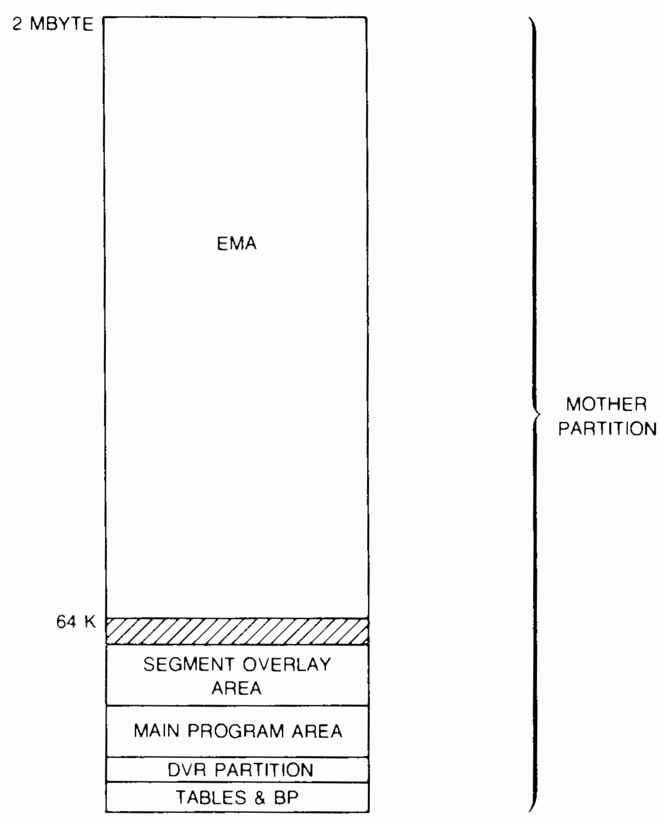


Figure 2

AN ANALOGY

The EMA data area can be looked at as a secondary data storage area, i.e., very much like disc storage. As such, data in EMA is not directly addressable but must first be brought into the logical address space of the program, i.e., small chunks of it are brought into MSEG. Here again the disc analogy holds because when we want to read/write a data item from/to a record, "chunks" of data (1 or more) are brought into main memory. However, for the FORTRAN programmer using EMA, all of that is TRANSPARENT. He can address 2M bytes of data.

MOTHER PARTITION

A partition that is larger than the maximum logical address space is called a "mother partition". A mother partition allows for subpartitions. RTE-IV will use mother partitions to dispatch programs that use an Extender Memory Area (EMA). Subpartitions of a mother partition have the same characteristics (real time or background) as the mother partition; they allow the user the capacity of using the large amount of memory belonging to the mother partition to run many smaller programs, when the mother partition is not in use.

For a more in-depth description of EMA, read the RTE-IV Programmers Reference Manual (92067-90001) and Wong and Manley, "RTE-IV: The Megaword-Array Operating System" in Hewlett-Packard Journal, October, 1978.

SHARING EXTENDED MEMORY ARRAYS IN RTE IV

Martha Robrahn/HP Neely Los Angeles

Once you have heard about all the wonderful things that Extended Memory Arrays (EMA) can do for you in RTE-IV, your next question is "Can I share EMA between programs?". HP's answer is no. However, with a few contributed subroutine calls and a little bit of overhead in the sharing programs, the average FORTRAN programmer can indeed share EMA between programs.

For those readers unfamiliar with the concept, EMA is a feature of the RTE-IV operating system that allows the programmer to access data arrays outside of his logical 32K word address space. This is accomplished at the program level by inserting one additional control statement into the source code. The statement has the form

$EMA(blockname,MSEG size)

where blockname is a labeled common block name and MSEG size is the number of pages of EMA mapped into the program's address space at one time. The MSEG size is essentially a movable "window" into the EMA area and is re-mapped by the EMA utility routines .EMAP, .EMIO and MMAP as required by the program. The minimum MSEG size is two pages. This size can be defaulted to 32 - program size - 1, the largest possible MSEG size, by setting the MSEG size to zero. Beyond the addition of this control statement, extended memory access of local common block (blockname) is completely transparent to the FORTRAN programmer.

EDITOR'S NOTE: Refer to the previous article for a full description of EMA.

For the purposes of this discussion, the following program names will be used: SEMA1 will designate the EMA program whose EMA is to be shared and SEMA2 will designate the program(s) which will share SEMA1's EMA.

To fully understand how to share EMA, one must first be familiar with the information in an EMA program's ID Segment and ID Segment Extension. (See figures 7 and 8)

At first glance it appears that if we modify WORD 2 of SEMA2's Segment Extension to reflect the physical starting page of SEMA1's EMA then we should be able to "fool" the EMA utility routines and reference SEMA1's EMA from SEMA2. In fact this approach does work, provided that both programs' EMA and COMMON declarations are identical.

```
FTN4,L
$EMA(BLK,0)
      PROGRAM SEMA1
      COMMON/BLK/I(30000),A(300,100)
C     .
C     .
C     .
      END
EMA SIZE=88 PAGES
```

Figure 1

```
FNT4,L
$EMA (BLK,0)
      PROGRAM SEMA2
      COMMON/BLK/I(30000),A(300,100)
C     .
C     .
C     .
      END
EMA SIZE=88 PAGES
```

Figure 2

However, this means that for a large EMA, the memory allocated to SEMA2's EMA (88 pages) will be unused. For most users, this approach is impractical to say the least.

Ideally one would like SEMA2 to have the capbilities of an EMA program with a minimum of wasted memory and associated overhead.

```
FTN4,L
$EMA(BLK,0)
      PROGRAM SEMA1
      COMMON/BLK/I(30000),A(300,100)
 C    .
 C    .
 C    .
      END
EMA SIZE=88 PAGES
```

Figure 3

```
FTN4,L
$EMA(BLK,2)
      PROGRAM SEMA2
      COMMON/BLK/I(1),A(1,1)
C     .
C     .
C     .
      A(300,100)=Q
      END
EMA SIZE=1 PAGE
```

Figure 4

This approach requires a little bit more work as well as a better understanding of how EMA works.

When an EMA program is compiled, FTN4 generates calls to the utility subroutine .EMAP for every reference made to an EMA variable. One of the parameters .EMAP uses is a table describing the variable referenced. For the arrays I and A in figure 3 above, these tables would be:

```
TABLI DEC 1         (# of dimensions)
      DEC -1        (negative of lower bound of demension 1)
      DEC 1         (# of words per element)
      DEC 0         (first of two word integer specifying the array's
                     offset from start of BLK - this is bits 15-0)
      DEC 0         (bits 31-16 of above)


TABLA DEC 2         (# of dimensions)
      DEC -1        (negative lower bound of dimension 2)
      DEC 300       (# of elements in 1st dimension)
      DEC -1        (negative lower bound of dimension 1)
      DEC 2         (# of words per element)
      DEC 30000     (first of two word integer specifying the array's
                     offset from start of BLK - this is bits 15-0)
      DEC 0         (bits 31-16 of above)
```

Figure 5

For the arrays I and A in figure 4, these tables would be:

```
TABLI DEC 1      (# of dimensions)
      DEC -1     (negative of lower bound of demension 1)
      DEC 1      (# of words per element)
      DEC 0      (first of two word integer specifying the array's
                  offset from start of BLK - this is bits 15-0)
      DEC 0      (bits 31-16 of above)


TABLA DEC 2      (# of dimensions)
      DEC -1     (negative lower bound of dimension 2)
      DEC 1      (# of elements in 1st dimension)
      DEC -1     (negative lower bound of dimension 1)
      DEC 2      (# of words per element)
      DEC 1      (first of two word integer specifying the array's
                  offset from start of BLK - this is bits 15-0)
      DEC 0      (bits 31-16 of above)
```

Figure 6

Every array and variable declared in EMA will have a similar table built by the compiler to be used by .EMAP. (The length of each table = 3 + 2 * [# of dimensions in the array].)

In order for SEMA2 in figure 4 to reference elements in array A beyond A(1,1), the EMA table for A in figure 6 must be modified to reflect larger dimensions and a different offset (i.e., to be the same as the EMA table for A in figure 5). In addition, word 28 of SEMA2's ID Segment must be modified to reflect a larger EMA size. Once these changes have been made to SEMA2, then it would be possible to access the entire array A in SEMA1 from SEMA2.

To put all these ideas into a cookbook type procedure (that really works), the following example is given for reader reference.

- Program SEMA1 (with the real EMA) runs and locks itself into memory.

- SEMA1 picks up it's ID Segment extension using subroutine **IDEX** and word 28 of the ID Segment using **EMASZ.** The calling sequences are as follows:

  ```
  CALL IDEX(IEXT)
  CALL EMASZ(ISIZE)
  ```

  where IEXT is a three element array, and word 28 of the user's ID segment is returned in ISIZE. These calls need only be issued once in each program.

- SEMA1 picks up the EMA tables corresponding to each array in EMA using **GETAB.** GETAB uses the following calling sequence:

  ```
  A(1,1)=0        (reference EMA variable)
  CALL GETAB(LEN,ITAB)
  ```

  Where A is the EMA array, LEN is the length of the table (3 + 2 * [# of dimensions]) and ITAB will contain the tables address on return.

  The subroutine GETAB finds the position of the EMA table for each array by picking up the calling address and searching backward in memory until it finds a JSB .EMAP. When this call is located, the subroutine picks up the EMA table address from the call and accesses the table using this address. Refer to the listing of GETAB to see this.

- SEMA1 schedules SEMA2 (and any other programs which will be sharing EMA) passing it the ID Segment word 28, ID Segment extension and EMA table information.

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--+
| List Linkage                                   |  Word  0  \
|------------------------------------------------|            |
| TEMP 1                                         |        1   |
| TEMP 2                                         |        2   |
| TEMP 3                                         |        3   |
| TEMP 4                                         |        4   |
| TEMP 5                                         |        5   |
|------------------------------------------------|            |
| Priority                                       |        6   |
| Primary Entry Point                            |        7 *|
|------------------------------------------------|            |
| Point of Suspension                            |        8   |
| A-Register                                     |        9   |
| B-Register                                     |       10   |
| EO-Registers                                   |       11   |
|------------------------|-----------------------|            | Memory
| Name 1                 | Name 2                |       12 *\ Resident
| Name 2                 | Name 4                |       13 */ Programs
|                        |--|--|--|--|-----------|            |
| Name 3                 |TM|ML|//|SS| Type      |       14 *|
|--|--|--|--|--|--|--|--|--|--|--|--|-----------|            |
|NA|//|NP| W| A|//| O|//| R| D|/////| Status    |       15   |
|--|--|--|--|--|--|--|--|--|--|-----|-----------|            |
| Time List Linkage                              |       16   |
|--------|--|------------------------------------|            |
| RES    | T| Multiple                           |       17   |
|--------|--|------------------------------------|            |
| Low Order 16 Bits of Time                      |       18   |
|------------------------------------------------|            |
| High Order 16 Bits of Time                     |       19   |
|--|--|--|--|--|--|--|--|-----------------------|            |
|BA|FW| M|AT|RM|RE|PW|RN| Father ID Segment No. |       20   |
|--|--|--|--|--|--|--|--|--|--|-----------------|            |
|RP|#pgs. (no BP) | MPFI   |//| Partition No. -1|       21   |
|--|--------------|--------|--|-----------------|            |
| Low Main Address                               |       22 *|
|------------------------------------------------|            |
| High Main Address + 1                          |       23 *|
|------------------------------------------------|            |
| Low Base Page Address                          |       24 *|
|------------------------------------------------|            |
| High Base Page Address                         |       25 */
|--|-----------------------|---------------------|
|LU| Program: Track        | Sector              |       26 *
|--|-----------------------|---------------------|
|LU| Swap:    Track        | No. Tracks          |       27
|--|--------------|--------|---------------------|
|ID Extension No. | EMA Size                     |       28
|-----------------|------------------------------|
| High Address + 1 of Largest Segment            |       29
|------------------------------------------------|
| Reserved                                       |       30\
|------------------------------------------------|          \Memory
| Reserved                                       |       31
|------------------------------------------------|          /Residents
| Negative MTM LU number                         |       32/
|------------------------------------------------|

* = words used in short ID segments for program segments
```

Figure 7

```
+------------------------------------------------------------------+
|                                                                  |
|   15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0                |
|   +--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|-+                |
|   |  |                        |               |   Word 0         |
|   |NS|      Current MSEG No.  | # Pages MSEG|                    |
|   |--|------------|--|--------------|-------------|              |
|   | MSEG Start   |DE| (Physical) EMA Start      |   Word 1        |
|   | Page (logic.)|  | Page                      |                 |
|   |--|-----------|--|---------------------------|                 |
|   |/////////////////| # Tracks for EMA Swap      |   Word 2        |
|   |/////////////////|                            |                |
|   +-----------------|----------------------------+                |
|                                                                  |
|   where:                                                         |
|                                                                  |
|     NS = 0 if the MSEG is pointing to a standard segment of      |
|            the EMA (set up by .EMAP)                             |
|                                                                  |
|        = 1 if the MSEG is pointing to a non-standard segment     |
|            (set up by .EMIO or .EMAP)                            |
|                                                                  |
|     DE = 0 if the EMA size was specified by the user             |
|                                                                  |
|        = 1 if the EMA size is allowed to default to the          |
|             maximum size available to the system                 |
|                                                                  |
+------------------------------------------------------------------+
```

Figure 8

- From this point on, SEMA1 can execute normally. It should, however, have some way of determining the status of the other programs sharing EMA so that it does not terminate before they are done accessing the data in his partition. In this example, a value in the EMA is checked as a completion flag.

- SEMA2 must declare its EMA to have the same structure as SEMA1's EMA. It need not be the same size. (See second example above and attached listings as examples).

- SEMA2 must pick up the information passed from SEMA1 and use **IDEX, EXSET** and **SZEMA** to appropriately modify his ID Segment Extension and ID Segment Word 28. The calling sequence is as follows:

```
CALL IDEX(MEXT)
MEXT(2)=IOR(IAND(MEXT(2),176000B),IAND(IEXT(2),1777B))
CALL EXSET(MEXT(2))
ISIZE=IAND(ISIZE,1777B)
CALL SZEMA(ISIZE)
```

  where IEXT is from above and MEXT is a new three element array and ISIZE is the new EMA size from above. and ISIZE is the new EMA size. These calls need only be issued once in each program.

- SEMA2 must also use the subroutine **SETAB** to modify the EMA Array tables to look like those passed from the father. The calling sequence is as follows:

```
A(1,1)=0        (reference EMA variable)
CALL SETAB(LEN,ITAB)
```

  where LEN and ITAB are described above. SETAB finds the position of the EMA table for each array by using the same algorithm as GETAB.

  This routine must be done once for every array accessed by a given program or subroutine in the program.

- SEMA2 is now accessing SEMA1's EMA and can access the full array dimensions declared in SEMA1's COMMON statement. In this example, SEMA2 changes all values in SEMA1's EMA. SEMA2 also calls a subroutine which accesses EMA and sets the completion flag to be checked by SEMA1.

SPECIAL NOTES AND CONSIDERATIONS

- Listings are included for all software referenced in this article.
- SEMA1 and SEMA2 must be type 3 programs in order to access ID Segments directly. To make them type 4 programs, IDEX, EMASZ, EXSET, and SZEMA would have to be modified to use cross map loads and stores where appropriate.
- SEMA1 must lock itself into memory to insure that the EMA does not disappear by SEMA1 being swapped. SEMA1 must also cooperate with all other programs sharing its EMA so that it does not terminate before they are done accessing its EMA.
- SEMA2 must lock itself into memory to prevent the dispatcher from overlapping SEMA1's EMA area.
- SEMA2 will, in general, fit in a non-EMA partition. In order to have an EMA program run in a non-EMA partition, you must use the AS,nn command when you load the program.
- The subroutine SETAB must be called in SEMA2 and in every subroutine of SEMA2 for each EMA array accessed. (See the example subroutine SHARE). This is because the compiler generates a separate set of EMA tables for each subroutine compiled.
- Special note to assembly language programmers: The inplementation of EMA and its access is considerably different from the FORTRAN level. (Refer to the RTE-IV Programmers Referenced Manual.) You will not need to use GETAB and SETAB in an assembly language program since you can put in the correct tables for .EMAP and .EMIO explicitly.
- 21MX-M users should be aware that they must force .EMAP to remap SEMA2's EMA once the EMA tables have been changed. This is due to the fact that the software version of .EMAP will only remap when necessary. An error will occur is the remapping is not forced. A good rule of thumb would be to reference the last element in SEMA1's EMA. (See example comments in SEMA2 listing)

Following this example, the average FORTRAN programmer can indeed share Extended Memory Arrays with a minimum amount of overhead.

Special thanks are due to Jim Grimm for the search algorithm used by GETAB and SETAB.

EDITOR'S NOTE: Shared EMA is not an HP supported utility. HP cannot assume liability for the information in this article and cannot assume liability for system integrity when these routines are used.

```
PAGE 0002 #01                              2:32 PM  TUE.,  5  DEC., 1978

                       ASMB,L
      00000                   NAM IDEX
                              ENT IDEX
    *THIS ROUTINE RETURNS THE CALLING PROGRAM'S ID EXTENSION
    * CALLING SEQUENCE IS
    *       CALL IDEX(IEXT)
    *                        WHERE IEXT IS AN ARRAY DIMENSIONED 3
                              EXT .ENTR
      01645            XIDEX EQU 1645B
      00000 000000     PTR   BSS 1
      00001 000000     IDEXT BSS 1
      00002 000000     IDEX  NOP
      00003 016001X           JSB .ENTR      PICK UP PARAMETER
      00004 000001R           DEF IDEXT
      00005 061645            LDA XIDEX      PICK UP ID EXTENSION
      00006 072000R           STA PTR
      00007 162000R           LDA PTR,I      AND PASS BACK TO CALLING PROGRAM
      00010 172001R           STA IDEXT,I
      00011 036001R           ISZ IDEXT
      00012 036000R           ISZ PTR
      00013 162000R           LDA PTR,I
      00014 172001R           STA IDEXT,I
      00015 036001R           ISZ IDEXT
      00016 036000R           ISZ PTR
      00017 162000R           LDA PTR,I
      00020 172001R           STA IDEXT,I
      00021 126002R           JMP IDEX,I
                              END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
PAGE 0002 #01                              2:32 PM  TUE.,  5  DEC., 1978

                       ASMB,L
      00000                   NAM EMASZ
                              ENT EMASZ
    * THIS ROUTINE RETURNS WORD 28 OF THE USER'S ID SEGMENT
    * THE CALLING SEQUENCE IS
    *       CALL EMASZ(IWORD)
                              EXT .ENTR
      00000 000000     IAD   BSS 1
      00001 000000     EMASZ NOP
      00002 016001X           JSB .ENTR      PICK UP PARAMETER
      00003 000000R           DEF IAD
      00004 061717            LDA XEQT
      00005 042011R           ADA D28
      00006 160000            LDA A,I
      00007 172000R           STA IAD,I
      00010 126001R           JMP EMASZ,I
      01717            XEQT  EQU 1717B
      00000            A     EQU 0
      00011 000034     D28   DEC 28
                              END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
PAGE 0002 #01                      2:33 PM  TUE.,  5  DEC., 1978

                   ASMB,L
     00000                  NAM GETAB
                            ENT GETAB
*    THIS ROUTINE WILL RETREIVE THE EMA TABLE FOR A GIVEN ARRAY
*    THE CALLING SEQUENCE IS AS FOLLOWS:
*    FTN,L
*    $EMA(BLK,N)
*          PROGRAM ...
*          COMMON /BLK/...ARRAY(J,K)
*              .
*              .
*              .
*          ARRAY(1,1)=0.
*          CALL GETAB(LEN,ITAB)
*              .          WHERE LEN=3+# OF DIMENSIONS*2
*              .                ITAB IS WHERE EMA TABLE IS RETURNED
*              .
                            EXT .ENTR,.EMAP
     00000 000000  LEN      BSS 1
     00001 000000  ITAB     BSS 1
     00002 000000  GETAB    NOP
     00003 016001X          JSB .ENTR      RETREIVE PARAMETERS
     00004 000000R          DEF LEN
     00005 066002R          LDB GETAB      PICK UP CALLING POINT
     00006 046025R          ADB M5
     00007 160001  SERCH    LDA B,I        AND TRACE BACKWARDS
     00010 052026R          CPA JSB        LOOKING FOR A JSB TO .EMAP
     00011 026014R          JMP FOUND
     00012 046024R          ADB M1
     00013 026007R          JMP SERCH
     00014 046023R FOUND    ADB D3         PICK UP ADDRESS OF EMA TABLE
     00015 160001           LDA B,I
     00016 066001R          LDB ITAB       AND PASS BACK TO CALLING PROGRAM
     00017 105777           MVW LEN,I
     00020 100000R
     00021 000000
     00022 126002R          JMP GETAB,I
     00023 000003  D3       DEC 3
     00024 177777  M1       DEC -1
     00001         B        EQU 1
     00025 177773  M5       DEC -5
     00026 016002X JSB      JSB .EMAP
                            END
**   NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
PAGE 0002 #01                        2:33 PM  TUE.,  5  DEC., 1978

                     ASMB,L
      00000                  NAM EXSET
                             ENT EXSET
* THIS ROUTINE WILL MODIFY THE CALLING PROGRAM'S ID EXTENSION
* TO REFLECT A NEW PHYSICAL START OF EMA
* CALLING SEQUENCE IS AS FOLLOWS
*       CALL EXSET(IWORD)
*                           WHERE IWORD IS THE NEW WORD 2 OF THE ID EXTENSION
                             EXT .ENTR,$LIBR,$LIBX
      01645           XIDEX  EQU 1645B
      00000 000000    PTR    BSS 1
      00001 000000    MEXT   BSS 1
      00002 000000    EXSET  NOP
      00003 016001X          JSB .ENTR      PICK UP PARAMETER
      00004 000001R          DEF MEXT
      00005 061645           LDA XIDEX      PICK UP ID EXTENSION
      00006 072000R          STA PTR
      00007 036000R          ISZ PTR
      00010 016002X          JSB $LIBR      GO PRIVILEGED
      00011 000000           NOP
      00012 162001R          LDA MEXT,I     AND MODIFY ID EXTENSION WORD 2
      00013 172000R          STA PTR,I
      00014 016003X          JSB $LIBX
      00015 000002R          DEF EXSET
                             END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**

PAGE 0002 #01                        2:33 PM  TUE.,  5  DEC., 1978

                     ASMB,L
      00000                  NAM SZEMA
                             ENT SZEMA
* THIS ROUTINE WILL MODIFY THE CALLING PROGRAM'S ID SEGMENT
* WORD 28 TO REFLECT THE PASSED EMA SIZE
* CALLING SEQUENCE IS
*       CALL SZEMA(ISIZE)
*                           WHERE ISIZE IS THE NEW EMA SIZE
                             EXT .ENTR,$LIBR,$LIBX
      00000 000000    IAD    BSS 1
      00001 000000    SZEMA  NOP
      00002 016001X          JSB .ENTR      PICK UP PARAMETERS
      00003 000000R          DEF IAD
      00004 065717           LDB XEQT       PICK UP WORD 28 OF ID SEGMENT
      00005 046017R          ADB D28
      00006 160001           LDA B,I
      00007 012016R          AND MASK       AND MODIFY EMA SIZE
      00010 132000R          IOR IAD,I
      00011 016002X          JSB $LIBR      GO PRIVELEGED
      00012 000000           NOP
      00013 170001           STA B,I        AND MODIFY IDSEG WORD 28
      00014 016003X          JSB $LIBX
      00015 000001R          DEF SZEMA
      01717           XEQT   EQU 1717B
      00001           B      EQU 1
      00016 176000    MASK   OCT 176000
      00017 000034    D28    DEC 28
                             END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
PAGE 0002 #01                                2:33 PM  TUE.,  5  DEC., 1978

                          ASMB,L
      00000                     NAM SETAB
                                ENT SETAB
*  THIS ROUTINE WILL OVERWRITE THE EMA TABLE FOR A GIVEN ARRAY
*  THE CALLING SEQUENCE IS AS FOLLOWS:
*  FTN,L
*  $EMA(BLK,N)
*       PROGRAM ...
*       COMMON /BLK/...ARRAY(J,K)
*           .
*           .
*           .
*       ARRAY(1,1)=0.
*       CALL SETAB(LEN,ITAB)
*           .             WHERE LEN=3+# OF DIMENSIONS*2
*           .                   ITAB IS THE NEW EMA TABLE
*           .
                                EXT .ENTR,.EMAP
      00000 000000  LEN   BSS 1
      00001 000000  ITAB  BSS 1
      00002 000000  SETAB NOP
      00003 016001X       JSB .ENTR      PICK UP PARAMETERS
      00004 000000R       DEF LEN
      00005 066002R       LDB SETAB      PICK UP CALLING LOCATION
      00006 046025R       ADB M5
      00007 160001  SERCH LDA B,I        AND TRACE BACKWARDS
      00010 052026R       CPA JSB        LOOKING FOR A JSB .EMAP
      00011 026014R       JMP FOUND
      00012 046024R       ADB M1
      00013 026007R       JMP SERCH
      00014 046023R FOUND ADB D3         PICK UP TABLE ADDRESS
      00015 164001        LDB B,I
      00016 062001R       LDA ITAB       AND OVERWRITE EMA TABLE
      00017 105777        MVW LEN,I
      00020 100000R
      00021 000000
      00022 126002R       JMP SETAB,I
      00023 000003  D3    DEC 3
      00024 177777  M1    DEC -1
      00001         B     EQU 1
      00025 177773  M5    DEC -5
      00026 016002X JSB   JSB .EMAP
                          END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
PAGE 0001  FTN.   2:21 PM  TUE.,  5  DEC., 1978

FTN,L
$EMA(BLK,0)
          PROGRAM SEMA1
          COMMON /BLK/IRAY(30000),X(3000,5),A(200,2,2)
          DIMENSION IMAP(32),IEXT(3),NAM(3),ITAB(10,4)
          DATA NAM/2HSE,2HMA,2H2 /
          LU=LOGLU(ID)
C    LOCK EMA ARRAYS INTO MEMORY
          CALL EXEC(22,1)
C    PICK UP INFORMATION ON EMA SIZE AND STARTING PAGE #
          CALL IDEX(IEXT)
          CALL EMASZ(ISIZE)
          ISIZE=IAND(ISIZE,1777B)
          WRITE(LU,16)IEXT,ISIZE
16        FORMAT(" ID EXTENSION IS "3(2X,O6) " EMA SIZE IS "I6)
C    PICK UP EMA TABLES
          ITAB(10,1)=3+1*2
          IRAY(1)=0
          CALL GETAB(ITAB(10,1),ITAB(1,1))
          ITAB(10,2)=3+2*2
          X(1,1)=0.
          CALL GETAB(ITAB(10,2),ITAB(1,2))
          ITAB(10,3)=3+3*2
          A(1,1,1)=0.
          CALL GETAB(ITAB(10,3),ITAB(1,3))
          LEN=30
C    PRESET IRAY
          DO 17 J=1,30000
17        IRAY(J)=30000-J
C    SCHEDULE SON AND PASS EMA INFORMATION
          CALL EXEC(24,NAM,IEXT(1),IEXT(2),IEXT(3),ISIZE,0,ITAB,LEN)
C    WAIT FOR SON TO SET FLAG
20        CALL EXEC(12,0,2,0,-5)
          IF(IRAY(25000).NE.0)GOTO20
40        WRITE(LU,30)(IRAY(J),J=1,10),(IRAY(J),J=24991,25000)
         &,X(3000,5),A(200,2,2)
C    CHECK FOR TASK COMPLETION
          IF(A(200,2,2).NE.-9999.)GOTO20
30        FORMAT(5I10/5I10/5I10/5I10/2F10.0)
          END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00471      COMMON = 00000
```

```
PAGE 0001  FTN.   2:21 PM  TUE.,  5  DEC., 1978

FTN,L
$EMA(BLK,2)
      PROGRAM SEMA2
      DIMENSION IEXT(5),MEXT(3),ITAB(10,4)
      COMMON/BLK/IRAY(1),X(1,1),A(1,1,1)
      EQUIVALENCE(ISTRT,MEXT(2)),(ISIZE,IEXT(4))
C  PICK UP EMA INFORMATION FROM FATHER
      CALL RMPAR(IEXT)
      CALL EXEC(14,1,ITAB,40)
      CALL ABREG(IA,IB)
      LEN=IB
C  PICK UP CURRENT ID EXTENSION
      CALL IDEX(MEXT)
      LU=LOGLU(ID)
      WRITE(LU,1)(ITAB(K),K=1,LEN)
1     FORMAT(" TABLE IS" 10(8(2X,O6)/))
      WRITE(LU,3)MEXT
3     FORMAT(" SEMA2 ID EXT "3(2X,O6))
C  MASK IN FATHER'S STARTING PAGE # OF EMA
      ISTRT=IOR(IAND(ISTRT,176000B),IAND(IEXT(2),1777B))
C  CALL EXSET TO MODIFY ID EXTENSION
      CALL EXSET(MEXT(2))
C  MODIFY EMA TABLES
      IRAY(1)=0
      CALL SETAB(ITAB(10,1),ITAB(1,1))
      X(1,1)=0.
      CALL SETAB(ITAB(10,2),ITAB(1,2))
      A(1,1,1)=0.
      CALL SETAB(ITAB(10,3),ITAB(1,3))
C  MODIFY EMA SIZE IN ID SEGMENT
      CALL SZEMA(ISIZE)
      WRITE(LU,3)MEXT
      WRITE(LU,5)
5     FORMAT(" WRITING TO EMA NOW")
C  PUT IN THIS TYPE OF CALL IF YOU HAVE AN MX-M
C     A(200,2,2)=0.
C  MAKE APPROPRIATE EMA CHANGES
      DO 10 J=1,30000
10    IRAY(J)=0
      DO 11 J=1,3000
      DO 11 K=1,5
11    X(J,K)=J*K
      DO 12 J=1,200
      DO 12 K=1,2
      DO 12 N=1,2
12    A(J,K,N)=J/K*N
      WRITE(LU,15)
15    FORMAT(" COMPLETED")
C  WAIT FOR FATHER TO VERIFY CHANGES
      CALL EXEC(12,0,3,0,-5)
C  CALL SUBROUTINE TO SET COMPLETION FLAG
      CALL SHARE(ITAB)
      END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00448      COMMON = 00000
```

```
PAGE 0001  FTN.   2:22 PM  TUE.,  5  DEC., 1978

FTN,L
$EMA(BLK,2)
      SUBROUTINE SHARE(ITAB)
      COMMON/BLK/IRAY(1),X(1,1),A(1,1,1)
      DIMENSION ITAB(10,4)
      A(1,1,1)=0.
      CALL SETAB(ITAB(10,3),ITAB(1,3))
      A(1,1,1)=-9999.
      A(200,2,2)=-9999.
      END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00082      COMMON = 00000
```

SHARED EMA FOR RTE-IV

Larry W. Smith/HP Fullerton

That's right! Shared-EMA is not only possible but extremely practical. How many of you could make use of the capability of sharing among several executing programs in real-time as much memory as you desire? Well, this can now be implemented by having a user make some minor on-line adjustments. The purpose of this article is to give you a general description of this capability and how you can implement shared-EMA on your RTE-IV system.

You might keep in mind that the solution presented in this article will work for both firmware and software versions of EMA and has been coined SHEMA/1000 by the originator and author of this article.

As it turns out, there are about two known methods of implemeting a shared-EMA capability without requiring a modification to operating system code and/or supported utilities. The method chosen for this article is the same that exists in an actual application.

SHEMA/1000 will soon be available through LOCUS (Library of Contributed User Software) for a nominal fee. Check future COMMUNICATOR issues for its announcement and part number or contact your local sales office.

THE APPLICATION

In many real-time applications, disc and memory usage are at a premium. The history of the development of SHEMA/1000 began with just such an application. The user had an application which put stringent demands on disc, input/output and common memory areas. The application involved controlling a radar control simulator and meant heavy interaction between terminals and external control devices, as well as continuous updating of transaction history files that recorded every trainee's reaction to simulated radar conditions. One fact seemed inescapable — the ability to share at least 45K words of memory by more than one program in real-time was a necessity. Conventional methods to solve this problem such as sharing disc and system common were carefully evaluated and not considered feasible due to existing performance limitations. Thus, a requirement for shared-EMA was born, carefully evaluated and considered to be the only time-wise solution.

THE PROBLEM

The ability to access a large data area in memory on a program basis was a significant enhancement to the RTE operating system. This capability gave the user easy reference to data arrays up to 915K words. Making this large data area shareable among programs by changing the firmware would be very difficult. A much better and overall effective solution would be to modify the appropriate area of system tables such that the normal operating system could be used.

An easy to use and flexible on-line solution was found that had four prime advantages to the user:

1. Required no disc or memory system code changes.
2. Did not degrade system performance assuming proper precautionary steps are taken.
3. Used standard means of declaring EMA at the source program level.
4. Any area of memory (declared or undeclared) could be shared.

In addition, program execution times remained relatively the same and the extra setup code in the participating programs is minimal.

In order to implement a shared-EMA scheme on-line, three major system problems had to be overcome:

PROBLEM 1: The scheduler and dispatcher had to be "fooled" initially as to a program's actual EMA requirements.

PROBLEM 2: Each participating program accessing the shared area of memory had to be setup to point to the proper physical page number of the shared-EMA area.

PROBLEM 3: The scheduler and dispatcher had to be "un-fooled" prior to execution of any normal EMA code.

These problem areas were all solved by manipulating information contained in a program's main ID segment and ID extension tables located in TABLE AREA II. Once this setup was done, all participating programs ran as usual in normal or mother partitions.

THE IMPLEMENTATION PROCEDURE

To implement SHEMA/1000, the following software modules were developed:

SHEMA — main program which is a pre-execution processor responsible for solving problem #1 AND #2.

IDMAP — FORTRAN callable assembly language subroutine which manipulates information in a programs main ID segment to solve problem #3.

IDEXT — FORTRAN callable assembly language subroutine which returns the address the ID extension table to help solve problem #1.

The functioning of SHEMA will be discussed later.

The overall flow of SHEMA/1000 is described in figure 1.

As an example, let's assume that the last 40K of memory is to be shared and declared as a mother partition. The partition layout for this particular system is described in figure 2.

The first step is to load SHEMA into the system. SHEMA can later be scheduled into any desired partition except a partition within the shared-EMA area. The next step (STEP 2) would be to load all participating programs with the subroutine "IDMAP" as you normally would by specifying all EMA requirements. Two test EMA programs, EMA1 and EMA2, were used to test SHEMA/1000. The first program initializes an entire 40K area of memory to consecutive integer values. The second program reads and verifies all integer values and prints an ending message

```
SHARED-EMA WORKS!
```

It is important at this point to note that any participating program cannot be placed into execution without SHEMA since the system will use its EMA requirements at load time to dispatch into a partition (partition 6 in the example).

Step 3 involves scheduling the program SHEMA once for each participating program. In our test case, we would have the following:

```
*RU,SHEMA,EMA1,170 -----> Initialized 40K word array.
        .
        .
  (wait until EMA1 completes execution)
        .
        .
*RU,SHEMA,EMA2,170 -----> Verifies 40K word array was initialized.
```

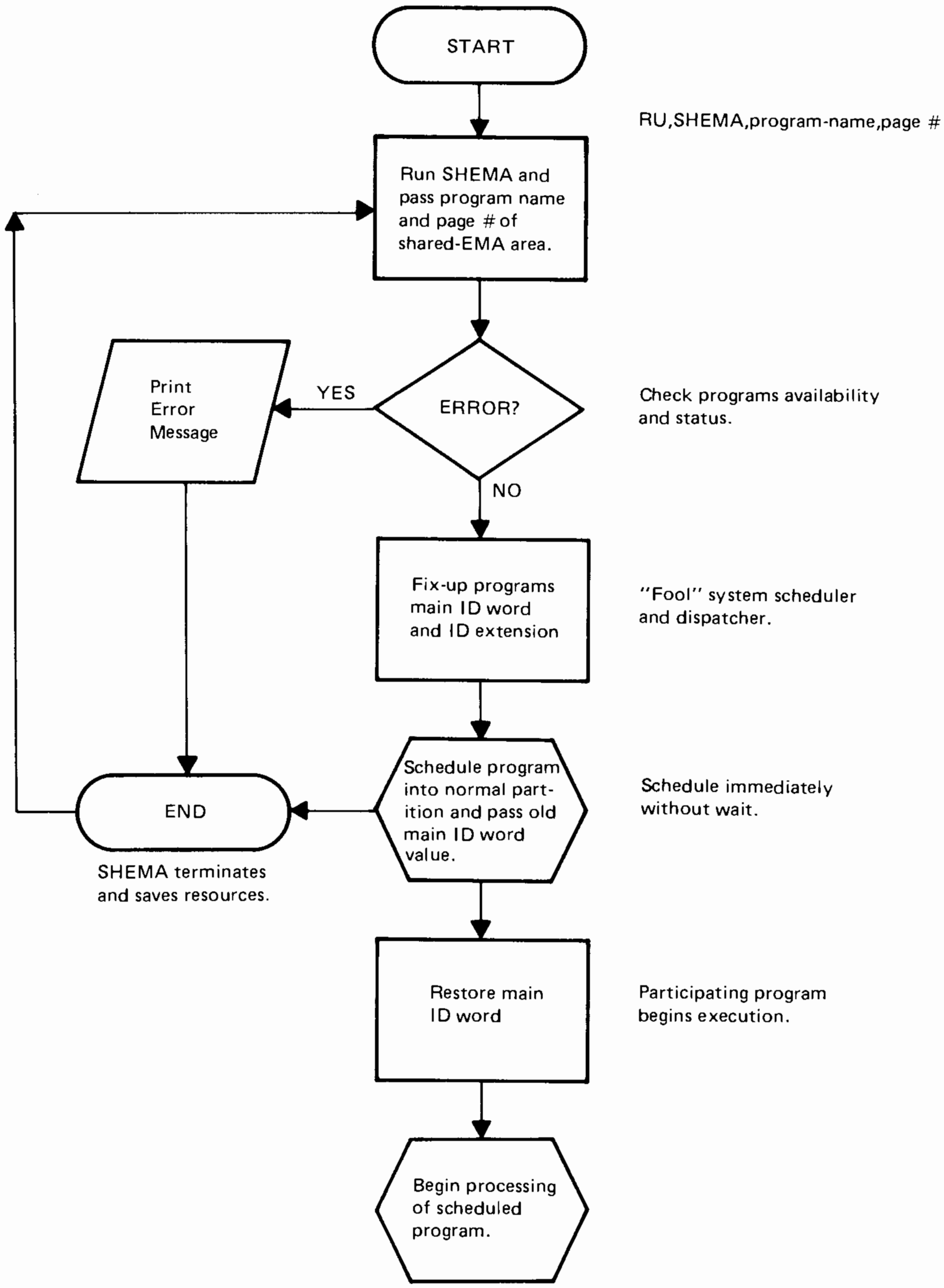

START
RU,SHEMA,program-name,page #
Run SHEMA and pass program name and page # of shared-EMA area.
ERROR?
Check programs availability and status.
YES
Print Error Message
NO
Fix-up programs main ID word and ID extension
"Fool" system scheduler and dispatcher.
Schedule program into normal partition and pass old main ID word value.
Schedule immediately without wait.
END
SHEMA terminates and saves resources.
Restore main ID word
Participating program begins execution.
Begin processing of scheduled program.

Figure 1

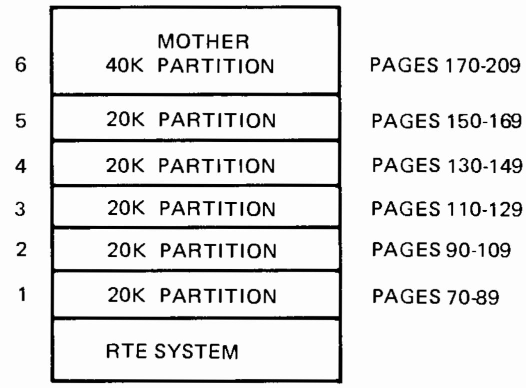


Figure 2

A program like SHEMA must first "fool" the system of the EMA requirements of programs EMA1 and EMA2 in the main ID segment and its corresponding ID extension. The ID extension is used later by the firmware for a page number. This is necessary so that EMA1 and EMA2 will not be forced into a mother partition if its total size exceeds the largest available normal partition. We must also put the starting physical page number (relative to 0) of the shared-EMA area into the programs ID extension before it is scheduled. This is necessary later so that the EMA firmware can properly construct the user's DMS map registers for actual memory access. After this is done, the program entered on the RU command is scheduled by SHEMA immediately without wait and passed the old contents of its main ID word. SHEMA then terminates and saves resources. It is now left up to the scheduled program to restore its EMA requirements in the main ID word before any EMA accesses are done. If this is not done, then the EMA firmware will default to standard array accessing methods (..MAP) which could cause disasterous and unpredictable results.

Each participating program is responsible for calling one subroutine to restore the main ID word before any access to any EMA variable is done. A skeleton of such a user program would look something like the following:

```
FTN4,L
$EMA(I,0)
        PROGRAM USER(3),THIS IS A SHARED-EMA TEST PROGRAM
        COMMON/I/IARY1(32000),IARY2(32000),IARY3(32000)
          .
          .
DIMENSION LUN(5)
          .
          .
CALL RMPAR(LUN)
          .
          .
CALL IDMAP(LUN)  -----> THIS IS THE ONLY SET-UP CODE.
          .
          .
< Begin normal code as usual >
          .
          .
END
END$
```

If the program wishes to terminate and re-execute at a later time or be put into the time-list, then it must save its main ID word, zero out the ID word, and restore when it re-executes.

The program SHEMA could be called a "shared-EMA pre-processor" since it does not actively participate in the EMA-sharing process during real-time. The flow of SHEMA described in figure 3.

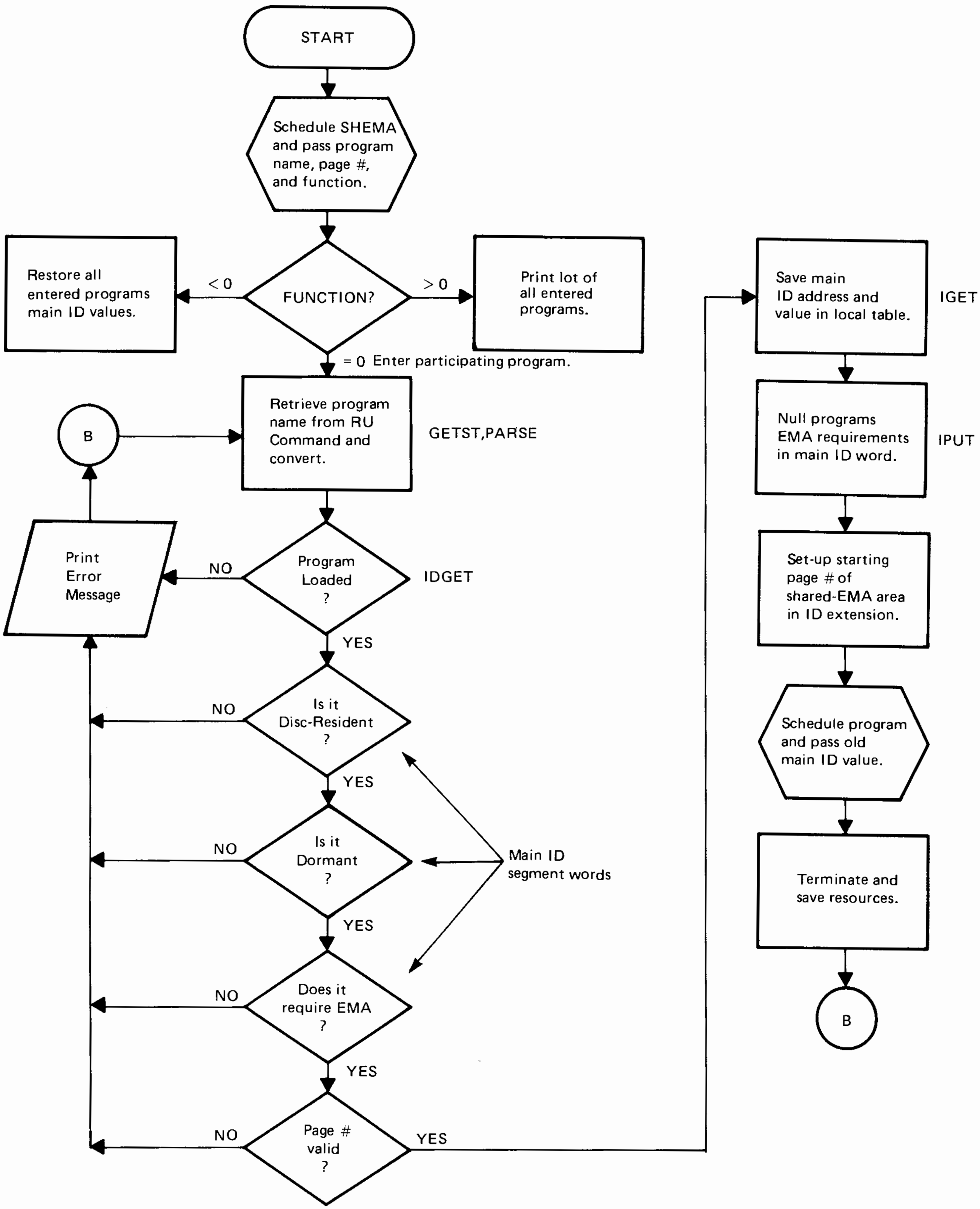

START
Schedule SHEMA and pass program name, page #, and function.
FUNCTION?
< 0
Restore all entered programs main ID values.
> 0
Print lot of all entered programs.
= 0 Enter participating program.
Retrieve program name from RU Command and convert.
GETST,PARSE
B
Program Loaded ?
IDGET
NO
Print Error Message
YES
Is it Disc-Resident ?
NO
YES
Is it Dormant ?
NO
YES
Does it require EMA ?
NO
YES
Page # valid ?
NO
YES
Main ID segment words
Save main ID address and value in local table.
IGET
Null programs EMA requirements in main ID word.
IPUT
Set-up starting page # of shared-EMA area in ID extension.
Schedule program and pass old main ID value.
Terminate and save resources.
B

Figure 3

THE LIMITATIONS

Although SHEMA/1000 or any shared-EMA scheme has a wide range of possible applications, the user must consider some limitations that could directly effect its implementation. Some limitations and cautions to consider are as follows:

1. All programs involved in shared EMA access must lock themselves into their partitions. Otherwise, as a program swaps in it will overlay the EMA area, which was possibly modified by other programs.

2. Although actual figures on system performance are not known at this point in time, it is not anticipated that performance will degrade significantly if SHEMA/1000 is used wisely. That is to say that a good mixture of compute-bound and I/O activity should be a consideration for maximum overall performance.

3. The usage of EMA variables create more object code (i.e. bigger program sizes) than non-EMA variables. Therefore, the use of EQUIVALENCE statements is suggested to reduce program loading and exectuion times.

4. Caution must be taken that a program's declared EMA size does not exceed the total amount of physical memory. If this is accidentally done, all accesses to variables within that area will result in a value of zero and the program will continue to run as normal.

5. If a participating program is loaded permanently into the system (i.e. with the LOADR command "OP,PE"), it must not be re-loaded without first restoring its main ID word. If this is not done, then its old ID extension will be lost forever or until the system is restored or re-generated.

CONCLUSION

I hope that you have found this article interesting and informative and and the solution given usable in your application.

I would like to extend many thanks to the personnel at Hughes Aircraft for their responsiveness and support given during the development and implementation of SHEMA/1000.

EDITOR'S NOTE: Shared EMA is not an HP supported utility. HP cannot assume liability for the information in this article and cannot assume liability for system integrity when these routines are used.

HOW TO BRING UP A DATA CAPTURE SYSTEM

Millo Fenzi & Paul Streit/HP Data Systems Division

INTRODUCTION

Data collection is a vital component of any manufacturing information system. Unfortunately, most currently available data capture techniques produce untimely and inaccurate data records. To solve these manufacturing related problems, Hewlett-Packard Company (HP) developed DATACAP/1000, a general purpose, real-time, manufacturing floor data capture software package. This article briefly describes the problems caused by the data capture system formerly used at one of HP's manufacturing divisions. It then describes the present system which uses DATACAP/1000 and an HP 1000 computer. The Manufacturing Division's application is discussed to outline the design and implementation phases that were required to bring up the data capture system.

BACKGROUND

Building 8 of Hewlett-Packard's Manufacturing Division contains a plastic and metal fabrication shop that makes components for other HP divisions. The shop contains plastic injection molding machines, aluminum diecasting devices, and sheetmetal forming equipment. A majority of the parts made by these machines have to be deburred and finished prior to their shipment. Different workers are responsible for these individual processes.

In order to accurately allocate labor cost, each worker kept a journal of the time spent working on a particular work order. Ten minutes before quitting time, workers would go through their journals and transfer the data; work order, runtime, operation, etc., to mark-sense cards. The next day, a secretary took the cards and read them through an optical card reader. This data was sent, via a leaseline hookup, to the corporate headquarters time share system. That evening the data was stripped off the timeshare system and put into a data base on the corporate mainframe computer.

PROBLEM AREAS

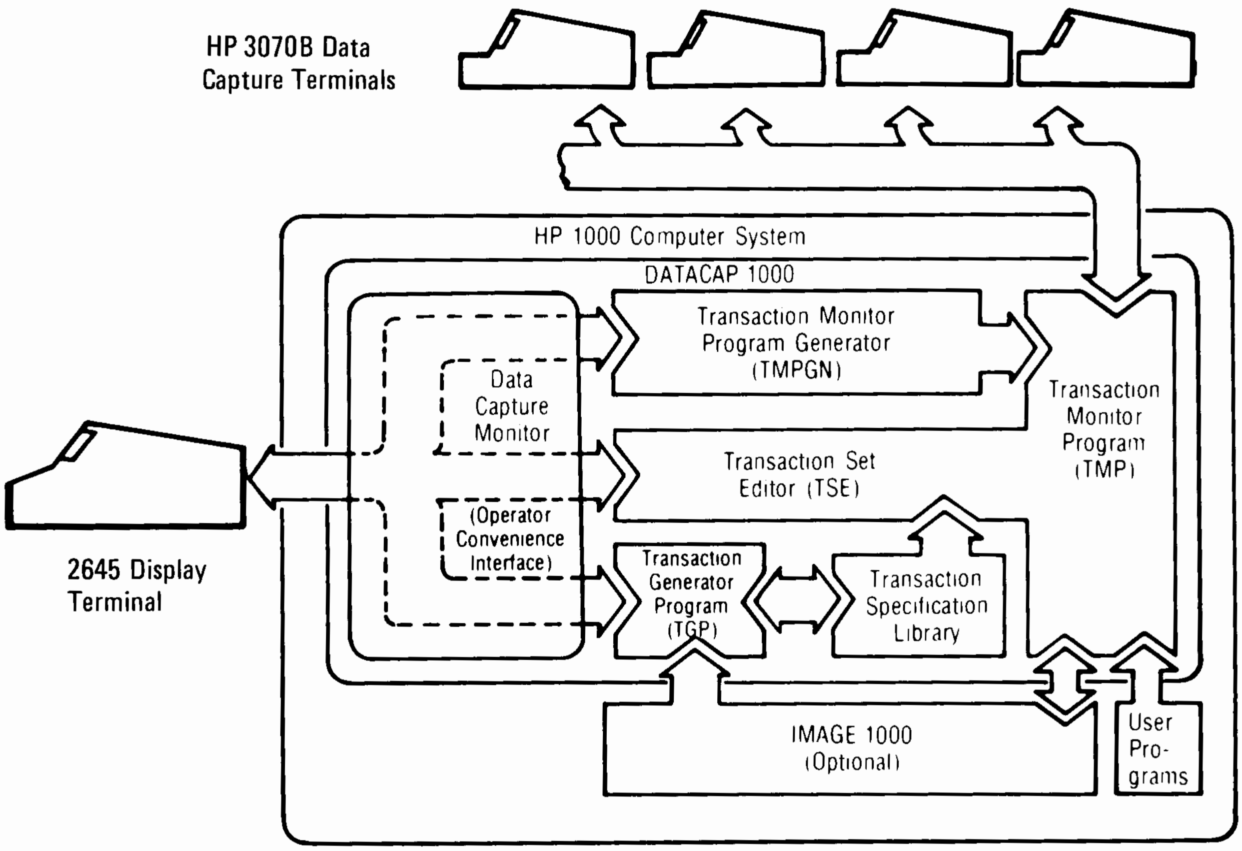
The major problem of this system was the time lag between document submission and error listing. There was possible turn around of five days! The workers who submitted documents often did not realize, because of this time lag, the connection between the document and the error message. Another problem was that a keypunch document could be handled by as many as nine people from the time it was filled out, entered into the corporate data base, and referred to the shop. This greatly increased the risk of keypunch errors and lost job vouchers. These two problems resulted in an inaccurate and non-current data base upon which the shop floor manager had to base his operating decisions.

In order to eliminate this non-current data base, a real-time factory floor data capture system was proposed. Ideally, the new system would immediately up-date the data base after first performing a series of validation operations on data entries. The new system should also be easily implemented and require a minimum of in-house software development. DATACAP/1000 and a HP 1000 computer system totally satisfies these requirements.

DATACAP DESCRIPTION

DATACAP/1000 is a complete data capture software package which is used in conjunction with an HP 1000 System and HP 307X data capture terminals. The software consists of three main programs, the first of which is the Transaction Monitor Program Generator (TMPGN). This program creates and loads into main memory a module which constantly monitors the data capture terminals. A second module, the Transaction Generator Program (TGP) leads the user through a series of screens on the system console. These screens precisely define the format and sequence in which the data must be entered on the 307X's for a particular transaction. This sequence is stored as a specific transaction and these transactions may be grouped into sets which are referred to as DATACAP libraries. The third module, the Transaction Set Editor (TSE) allows the user to selectively place individual transactions or libraries into a "working set". This "working set" contains the transactions that can be accessed from the data capture terminals via the monitoring program.

DATACAP/1000



DESIGNING A DATACAP SYSTEM

To implement a DATACAP/1000 system, one must complete four steps of a design cycle:

1. Determine what information you want and what data will give it to you.
2. Structure a data base around this data.
3. Determine how to most efficiently update the data base.
4. Design transactions to accomplish step 3.

Given that the system designer has followed these steps he/she is ready to create transactions with the TGP.

To create a transaction, the user needs to first identify the transaction by name and number. Next, special functions can be assigned to the ten programmable soft keys on the data capture terminals. The user is now ready to specify what questions, or data input prompts, the transaction should contain. Associated with each input are a series of specifications defining the data format; real, integer, string or data base item and form; card, badge or keyboard. Each data format has optional validation parameters e.g., upper and lower bounds or character masks, which can be applied to the inputs. In addition, the user can designate which prompting lights will be lit and what data is to be displayed on the data capture terminal prior to each input.

Once the data is entered, other information such as the date, time, and transaction number can be optionally added. All of this information can then be stored in either an IMAGE data base, serial disc file, magnetic tape, or any combination of the three.

As a result of the first step in the design cycle the system designer determined that the Manufacturing Division required two outputs. One was a job status report that kept track of work-in-process inventory. The second was a labor voucher report that allocated shop labor to individual work orders or account numbers and was used for costing, cost accounting, development of standard times, and the complete allocation of each workers' eight hour day.

The IMAGE data base schema that satisfied these requirements had three detail data sets, labor voucher to work order, labor voucher to account number, and job move information. Each of these detail sets contained data items specific to their information requirements. Four master data sets, work order number, location code, employee number and charge account, are also within the data base. These master sets are accessed during transactions for on-line data input validation. It was determined that the data should be entered by individual workers as they finished specific jobs and moved in-process inventory to the next location. This completes the four steps of the design cycle.

At this point, nine transactions were drawn up. Three of them account for labor, two for move orders, and the remaining four are for maintenance purposes. One of the three labor vouchering transactions charges labor to an account number and the other two to work orders. Two are needed to charge to work orders because one is for orders with official move cards whereas the other is for prototypes and specials which do not have cards. The two move order transactions are split the same way, one with and one without an official move card. The four maintenance transactions are designed for the shop floor supervisors to use when correcting incorrect or absent data records.

DATA CAPTURE TERMINAL OPERATION

To actually initiate one of these transactions, a worker walks up to one of the data capture terminals and hits an attention key. When the terminal indicates that it is ready, the worker keys in a transaction specification number. This number uniquely labels a transaction. Once a transaction has been specified, the terminal runs through a series of prompting lights which ask the worker to input data.

A typical transaction, e.g., labor charge to work orders, first prompts the worker for their shift number. The next prompt asks for the workers name which is entered by way of a punched type III plastic badge. Worker location and the quantity in the work order are the two following prompts. Both of these items are input via the keybaord. The next prompt is for the punched move card which contains eleven invariant data items associated with the work order. Finally, the lights prompt for run hours and setup hours, both of which are entered via the keybaord.

All of the keyboard inputs are subjected to validation procedures. The shift response is checked against a character mask. Location is verified by checking for existence in the data base. The quantity moved input is bounded by 0 and 10,000. Finally, the run and setup hours are bounded by 0 and 8.

An additional feature of the transaction occurs when the worker enters his/her name badge. At this point, the DATACAP retrieves that workers home location code from an IMAGE Database and displays it on the data capture terminal so that it can be entered by a single keystroke given that the worker is in his/her home location. This reduces keyboard input and thus reduces the amount of time required to complete a transaction. When the worker indicates that the transaction is completed four system data items of time, date, terminal and transaction numbers are logged with the data record as it is put into the data base.

HOW MANY TERMINALS?

Having completed the design cycle, we now face the practical task of implementing the system. How many data capture terminals are appropriate for it?

A first cut at this figure can be achieved by positioning one data capture terminal at the center of each work area. A work area is defined as a group of machines that perform common operations. The number of terminals actually required is dependent on how long it takes workers to complete transactions and how often they run transactions. The Manufacturing Division has 27 terminals for its 600 employees to use, or about 22 workers per terminal. Placement of the terminals is straight forward, they are located in the center of work areas equidistant from all potential users. The terminals are placed on pedestals and have racks for the employee badges and move cards mounted above them.

WORKING TRAINING

The data capture terminals are simple to operate and employees learn to use them in a matter of days. Typically, it is the line supervisors responsibility to train workers. A one week acclimatizaion period is provided during which the terminal is set up with dummy transactions and workers are free to walk up and familiarize themselves with the DATACAP concept. The following two days are spent with parallel terminal entry and mark sense cards. This redundance is used to check the accuracy of the workers' terminal entries. At this point, most operators are competent and normal terminal operation follows.

SYSTEM CONFIGURATION

The Manufacturing Division is responsible for the operations in three buildings, each a few miles apart. The various facilities required different length data links from the HP 1000 to the most distant data capture terminals. These links range from one to four thousand feet, well below the system maximum of over seven thousand feet.

Identical HP 1000 systems were installed at all three sites. The data is presently stored on magnetic tape and then nightly transfered to a central node. Within two months, a conversion to DS/1000 links is planned. At the central node the data is modified into a compatible format and then sent via a Remote Job Entry (RJE) phone connection to the corporate mainframe for data processing. On site, the data is used daily to give supervisors and managers timely and accurate information on work-in-process inventory, labor costs, and employee statistics.

CONCLUSION

DATACAP/1000 is a sophisticated software package and cannot simply be bought and plugged in. In order to bring up an efficient data capture system with a minimum of complications, a few simple steps must be followed. The intent behind this step-by-step description is to give other manufacturing companies the benefit of the Manufacturing Divisions' experience. First of all the design cycle must be completed. This causes the user to identify what, where, and how data must be captured. The design cycle results in a number of transaction specifications which can then be easily filled out on a console. The user must then determine how many data capture terminals are required throughout the facility. Workers must be trained to use the data capture terminals. Finally the data capture system must be tied into the existing data processing network. Upon completing these steps the user posseses a data capture system that collects timely and accurate data. This precise data is the basis for a stronger manufacturing information and control system.

Sample IMAGE schema for a data capture application

```
$CONTROL LIST,SET,TABLE,ROOT;
BEGIN DATA BASE BLDG8;CR022;32767;
LEVELS:
ITEMS:
    0001      MEMPNO,   U6;       << KEY MANUAL EMPLOYEE NO. >>
    0002      MHLOC,    U6;       << MANUAL HOME LOCATION >>
    0003      MORDNO,   U8;       << KEY AUTO WORK ORDER >>
    0004      MLOC,     U6;       << KEY MANUAL LOCATION CODE >>
    0005      MACCT,    U4;       << KEY MANUAL CHARGE TO ACCOUNT # >>
    0006      EMPNO,    U6;       << KEY DETAIL LABOR EMPL. NO. >>
    0007      CHLOC,    U6;       << KEY DETAIL LBR. CHARGE LOCN. >>
    0008      ORDNO,    U8;       << KEY DETAIL LABOR W.O. >>
    0009      ITMNO,    U2;       << MAPS\LABOR. ITEM NUMBER >>
    0010      PRTNO,    U14;      << MAPS\LBR. PART NUMBER >>
    0011      TRCDE,    U4;       << MAPS\LBR. TRANS. CODE\MFG. AREA >>
    0012      FAREA,    U2;       << MAPS\LBR. FROM AREA >>
    0013      DIV,      U2;       << MAPS\LBR. DIVISION NO. >>
    0014      FSEQ,     U4;       << MAPS\LBR. FROM SEQUENCE >>
    0015      FSECT,    U4;       << MAPS\LBR. FROM SECTION >>
    0016      FWSTA,    U4;       << MAPS\LBR. FROM WORK STATION >>
    0017      SURUN,    U10;      << MAPS\LBR. STD. SETUP\RUN HOURS >>
    0018      TASWS,    U12;      << MAPS\LBR. TO AREA\SECT\W.STA\SEQ >>
    0019      ASU,      R2;       << LABOR ACTUAL SETUP HOURS >>
    0020      ARUN,     R2;       << LABOR ACTUAL RUN HOURS >>
    0021      QTY,      R2;       << LABOR QTY. SENT TO NEXT OPER. >>
    0022      SHFOT,    U2;       << SHIFT\OVERTIME CODE >>
    0023      VODTE,    U6;       << LABOR VOUCHERED DATE >>
    0024      TRID,     U4;       << LABOR TRANS. IDENT. >>
    0025      TMNO,     U2;       << LABOR TERMINAL NUMBER >>
    0026      TRDTE,    U6;       << LABOR TRANSACTION DATE >>
    0027      TRTIM,    U4;       << LABOR TRANSACTION TIME >>
    0028      ORDNO1,   U8;       << KEY DETAIL MAPS MOVE W.O. >>
    0029      ITMNO1,   U2;       << MAPS ITEM NUMBER >>
    0030      PRTNO1,   U14;      << MAPS PART NUMBER >>
    0031      TRCDE1,   U4;       << MAPS TRANS. CODE\MFG. AREA >>
    0032      FAREA1,   U2;       << MAPS FROM AREA >>
    0033      DIV1,     U2;       << MAPS DIVISION NUMBER >>
    0034      FSEQ1,    U4;       << MAPS FROM SEQUENCE NO. >>
    0035      FSECT1,   U4;       << MAPS FROM SECTION >>
    0036      FWSTA1,   U4;       << MAPS FROM WORK STATION >>
    0037      SURUN1,   U10;      << MAPS STD. SETUP\RUN HOURS >>
    0038      TASWS1,   U12;      << MAPS TO AREA\SECT\W.STA\SEQ >>
    0039      PCCDE1,   U2;       << PARTIAL\COMPLETE MOVE CODE >>
    0040      QTY1,     R2;       << QTY. SENT TO NEXT OPERATION >>
    0041      TRID1,    U4;       << MAPS TRANSACTION IDENT. >>
    0042      TMNO1,    U2;       << MAPS TERMINAL NUMBER >>
    0043      TRDTE1,   U6;       << MAPS TRANSACTION DATE >>
    0044      TRTIM1,   U4;       << MAPS TRANSACTION TIME >>
    0045      EMPNO2,   U6;       << KEY DETAIL LABOR EMPL. NO. >>
    0046      CHLOC2,   U6;       << KEY DETAIL LBR. CHARGE LOCN. >>
    0047      ACCT2,    U4;       << KEY DETAIL LABOR ACCOUNT >>
    0048      FAREA2,   U2;       << LABOR FROM AREA >>
    0049      FWSTA2,   U4;       << LABOR FROM WORK STATION >>
    0050      ASU2,     R2;       << LABOR ACTUAL SETUP HOURS >>
    0051      ARUN2,    R2;       << LABOR ACTUAL RUN HOURS >>
    0052      QTY2,     R2;       << LABOR QTY. SENT TO NEXT OPER. >>
    0053      SHFOT2,   U2;       << SHIFT\OVERTIME CODE >>
    0054      VODTE2,   U6;       << LABOR VOUCHERED DATE >>
    0055      TRID2,    U4;       << LABOR TRANS. IDENT. >>
    0056      TMNO2,    U2;       << LABOR TERMINAL NUMBER >>
    0057      TRDTE2,   U6;       << LABOR TRANSACTION DATE >>
    0058      TRTIM2,   U4;       << LABOR TRANSACTION TIME >>
```

```
SETS:
         NAME:      EMPFL,M,CR022;       << MANUAL MASTER EMPLOYEE NO. >>
         ENTRY:     MEMPNO(2),
                    MHLOC;
         CAPACITY:  1301;

         NAME:      ORDFL,A,CR022;       << MANUAL MASTER WORK ORDER >>
         ENTRY:     MORDNO(2);
         CAPACITY:  10007;

         NAME:      LOCFL,M,CR022;       << MANUAL MASTER LOCATION CODE >>
         ENTRY:     MLOC(2);
         CAPACITY:  199;

         NAME:      ACCFL,M,CR022;       << MANUAL MASTER CHARGE ACCT. >>
         ENTRY:     MACCT(1);
         CAPACITY:  199;

         NAME:      LWOFL,D,CR022;       << DETAIL LABOR TO W. O. >>
         ENTRY:     EMPNO(EMPFL),
                    CHLOC(LOCFL),
                    ORDNO(ORDFL),
                    ITMNO,
                    PRTNO,
                    TRCDE,
                    FAREA,
                    DIV,
                    FSEQ,
                    FSECT,
                    FWSTA,
                    SURUN,
                    TASWS,
                    ASU,
                    ARUN,
                    QTY,
                    SHFOT,
                    VODTE,
                    TRID,
                    TMNO,
                    TRDTE,
                    TRTIM;
         CAPACITY:  8011;
```

```
        NAME:      MAPFL,D,CR022;          << DETAIL MAPS MOVE INFO. >>
        ENTRY:     ORDNO1(ORDFL),
                   ITMNO1,
                   PRTNO1,
                   TRCDE1,
                   FAREA1,
                   DIV1,
                   FSEQ1,
                   FSECT1,
                   FWSTA1,
                   SURUN1,
                   TASWS1,
                   PCCDE1,
                   QTY1,
                   TRID1,
                   TMNO1,
                   TRDTE1,
                   TRTIM1;
        CAPACITY:  7499;

        NAME:      LACFL,D,CR022;          << DETAIL LABOR TO ACCOUNT >>
        ENTRY:     EMPNO2(EMPFL),
                   CHLOC2(LOCFL),
                   ACCT2(ACCFL),
                   FAREA2,
                   FWSTA2,
                   ASU2,
                   ARUN2,
                   QTY2,
                   SHFOT2,
                   VODTE2,
                   TRID2,
                   TMNO2,
                   TRDTE2,
                   TRTIM2;
        CAPACITY:  1511;
END.
```

DATA SET NAME	TYPE	FLD CNT	PATH CNT	ENTR LGTH	MED REC	CAPAC CT	CART NO.
EMPFL	M	2	2	6	9	1301	CR022
ORDFL	A	1	2	4	9	10007	CR022
LOCFL	M	1	2	3	9	199	CR022
ACCFL	M	1	1	2	6	199	CR022
LWOFL	D	22	3	57	7	8011	CR022
MAPFL	D	17	1	44	3	7499	CR022
LACFL	D	14	3	29	7	1511	CR022

```
NUMBER OF ERROR MESSAGES    0
ITEM NAME COUNT:  58
DATA SET COUNT:  7
ROOT LENGTH:    5 BLOCKS, 544 WORDS
```

CARTRIDGE REFERENCE NUMBER	NUMBER OF BLOCKS REQ'D
CR022	8387.

Sample DATACAP transaction specification

```
HP3070B LABEL PRINTOUT FOR TRANSACTION SPECIFICATION : DCS06  / 6

                FROM TRANSACTION SPECIFICATION LIBRARY : CTU41  (CR =     2)

*****************************************************************************
*              *              *              *              *              *
*      []      *      []      *      []      *      []      *      []      *
*              *              *              *              *              *
*              *              *              *              *              *
* SHIFT/O.T.?  * EMPL. BADGE? * CHARGE LOC.? *              *   ERROR !    *
*              *              *              *              *              *
*****************************************************************************
*              *              *              *              *              *
*      []      *      []      *      []      *      []      *      []      *
*              *              *              *              *              *
*              *              *              *              *              *
* QTY. MOVED?  * MOVE CARD?   *              *              * COMPLETE TR. *
*              *              *              *              *              *
*****************************************************************************
*              *              *              *              *              *
*      []      *      []      *      []      *      []      *      []      *
*              *              *              *              *              *
*              *              *              *              *              *
*              *              * SETUP HOURS? * RUN HOURS?   * SPEC#-[SC] ? *
*              *              *              *              *              *
*****************************************************************************
*              *              *              *              *              *
* ============ * ============ * ============ * ============ * ============ *
* ============ * ============ * ============ * ============ * ============ *
*              *              *              *              *              *
*              *              *              *              *              *
* ABORT/SELECT *              *              *              * TR. COMPLETE *
*              *              *              *              *              *
*****************************************************************************
*              *              *              *              *              *
* ============ * ============ * ============ * ============ * ============ *
* ============ * ============ * ============ * ============ * ============ *
*              *              *              *              *              *
*              *              *              *              *              *
*              *              *              *              *              *
*              *              *              *              *              *
*              *              *              *              *              *
*****************************************************************************

   THIS TRANSACTION USES FEATURES AVAILABLE ONLY ON THE HP3070B, IE,

        CARD READER
        PRINTER

   THEREFORE NO HP3070A LABEL MODEL IS PROVIDED
```

```
                    TRANSACTION SPECIFICATION DOCUMENTATION
                    ***************************************

                         SYSTEM DATE    : 12- 6-1978
                         FROM LIBRARY   :  CTU41  (CR =     2)

                         NAME           :  DCS06
                         NUMBER         :  6
                         SECURITY CODE  :

                        IMAGE DATA BASE : BLDG8

   SPECIAL FUNCTION KEYS ASSIGNMENT :
   *********************************

 KEY#    NORMAL VALUE/FUNCTION     PREFIXED VALUE/FUNCTION    TERMINATOR ?

  01          RESET
  02          ABORT/SELECT                                          YE
  03                                                                NO
  04                                                                NO
  05                                                                NO
  06          TR. COMPLETE                                          YE
  07                                                                NO
  08                                                                NO
  09                                                                NO
  10                                                                NO
  11                                                                NO


        17  U QUESTIONS :
        ****************

          QUESTION LABEL :  SHIFT/O.T.?
          ------------------------------

              - DISPLAYED INFORMATION :
                               LIGHT # : 1
                                  TYPE : STRING  (LENGTH =    2)
                        DISPLAY MODULE : DIS06
                           PRINT VALUE : YE
                  DATA OFFSET IN BUFFER :    9

              - ANSWER DEFINITION :
                                 INPUT : KEYBOARD
                               LIGHT # : 01
                                  TYPE : STRING  (LENGTH =    2)
                       IMAGE ITEM NAME : SHFOT   (FUNCTION : A)
                           POSITIONING : L
                                  MASK : 9X
                         DEFAULT VALUE :
                  DATA OFFSET IN BUFFER :   12
```

```
QUESTION LABEL :  EMPL. BADGE?
------------------------------

    - ANSWER DEFINITION :
                         INPUT : READER
                      NEW CARD : A.H.80
                       LIGHT # : 2
                          TYPE : STRING  (LENGTH =    6)
               IMAGE ITEM NAME : MEMPNO  (FUNCTION : F)
          IMAGE EDIT GENERATED : CHECK EXISTENCE
                   POSITIONING : L
                          MASK : 99999
                 DEFAULT VALUE :
         DATA OFFSET IN BUFFER :   11

QUESTION LABEL :  CHARGE LOC.?
------------------------------

    - DISPLAYED INFORMATION :
                       LIGHT # : 3
                          TYPE : STRING  (LENGTH =    6)
               IMAGE ITEM NAME : MHLOC
                   PRINT VALUE : YE
         DATA OFFSET IN BUFFER :   14

    - ANSWER DEFINITION :
                         INPUT : KEYBOARD
                       LIGHT # : 3
                          TYPE : STRING  (LENGTH =    6)
               IMAGE ITEM NAME : CHLOC   (FUNCTION : A)
          IMAGE EDIT GENERATED : CHECK EXISTENCE
                   POSITIONING : L
                          MASK : 999999
                 DEFAULT VALUE : DISPLAYED VALUE
         DATA OFFSET IN BUFFER :   17

QUESTION LABEL :  QTY. MOVED?
------------------------------

    - ANSWER DEFINITION :
                         INPUT : KEYBOARD
                       LIGHT # : 6
                          TYPE : REAL
               IMAGE ITEM NAME : QTY     (FUNCTION : A)
                   UPPER LIMIT : 99999
                   LOWER LIMIT : 0
                 DEFAULT VALUE : 0
         DATA OFFSET IN BUFFER :   20
```

```
QUESTION LABEL :  MOVE CARD?
------------------------------

    - ANSWER DEFINITION :
                         INPUT : READER
                      NEW CARD : A.H.80
                       LIGHT # : 7
                          TYPE : STRING  (LENGTH =   4)
               IMAGE ITEM NAME : TRCDE   (FUNCTION : A)
                   POSITIONING : L
                          MASK : \X99
                   EDIT MODULE : MVCRD
                 DEFAULT VALUE :
         DATA OFFSET IN BUFFER :   22

QUESTION LABEL :  FROM AREA
------------------------------

    - ANSWER DEFINITION :
                         INPUT : READER
                       LIGHT # : 7
                          TYPE : STRING  (LENGTH =   2)
               IMAGE ITEM NAME : FAREA   (FUNCTION : A)
                   POSITIONING : L
                          MASK : 99
                 DEFAULT VALUE :
         DATA OFFSET IN BUFFER :   24

QUESTION LABEL :  DIVISION #
------------------------------

    - ANSWER DEFINITION :
                         INPUT : READER
                       LIGHT # : 7
                          TYPE : STRING  (LENGTH =   2)
               IMAGE ITEM NAME : DIV     (FUNCTION : A)
                   POSITIONING : L
                          MASK : 99
                 DEFAULT VALUE :
         DATA OFFSET IN BUFFER :   25

QUESTION LABEL :  ORDER #
------------------------------

    - ANSWER DEFINITION :
                         INPUT : READER
                       LIGHT # : 7
                          TYPE : STRING  (LENGTH =   8)
               IMAGE ITEM NAME : ORDNO   (FUNCTION : A)
                   POSITIONING : L
                          MASK : 99999999
                 DEFAULT VALUE :
         DATA OFFSET IN BUFFER :   26
```

```
QUESTION LABEL :  ITEM #
------------------------------

    - ANSWER DEFINITION :
                            INPUT : READER
                          LIGHT # : 7
                             TYPE : STRING  (LENGTH =   2)
                  IMAGE ITEM NAME : ITMNO   (FUNCTION : A)
                      POSITIONING : L
                             MASK : 99
                    DEFAULT VALUE :
            DATA OFFSET IN BUFFER :   30

QUESTION LABEL :  SEQUENCE #
------------------------------

    - ANSWER DEFINITION :
                            INPUT : READER
                          LIGHT # : 7
                             TYPE : STRING  (LENGTH =   4)
                  IMAGE ITEM NAME : FSEQ    (FUNCTION : A)
                      POSITIONING : L
                    DEFAULT VALUE :
            DATA OFFSET IN BUFFER :   31

QUESTION LABEL :  PART #
------------------------------

    - ANSWER DEFINITION :
                            INPUT : READER
                          LIGHT # : 7
                             TYPE : STRING  (LENGTH =  14)
                  IMAGE ITEM NAME : PRTNO   (FUNCTION : A)
                      POSITIONING : L
                    DEFAULT VALUE :
            DATA OFFSET IN BUFFER :   33

QUESTION LABEL :  SECTION #
------------------------------

    - ANSWER DEFINITION :
                            INPUT : READER
                          LIGHT # : 7
                             TYPE : STRING  (LENGTH =   4)
                  IMAGE ITEM NAME : FSECT   (FUNCTION : A)
                      POSITIONING : L
                    DEFAULT VALUE :
            DATA OFFSET IN BUFFER :   40

QUESTION LABEL :  OPERATION #
------------------------------

    - ANSWER DEFINITION :
                            INPUT : READER
                          LIGHT # : 7
                             TYPE : STRING  (LENGTH =   4)
                  IMAGE ITEM NAME : FWSTA   (FUNCTION : A)
                      POSITIONING : L
                    DEFAULT VALUE :
            DATA OFFSET IN BUFFER :   42
```

```
QUESTION LABEL :  STD. SU/RUN
------------------------------

    - ANSWER DEFINITION :
                          INPUT : READER
                        LIGHT # : 7
                           TYPE : STRING  (LENGTH =  10)
                IMAGE ITEM NAME : SURUN   (FUNCTION : A)
                    POSITIONING : R
                  DEFAULT VALUE :
          DATA OFFSET IN BUFFER :   44

QUESTION LABEL :  TO A/S/WS/S
------------------------------

    - ANSWER DEFINITION :
                          INPUT : READER
                        LIGHT # : 7
                           TYPE : STRING  (LENGTH =  12)
                IMAGE ITEM NAME : TASWS   (FUNCTION : A)
                    POSITIONING : L
                  DEFAULT VALUE :
          DATA OFFSET IN BUFFER :   49

QUESTION LABEL :  SETUP HOURS?
------------------------------

    - ANSWER DEFINITION :
                          INPUT : KEYBOARD
                        LIGHT # : 13
                           TYPE : REAL
                IMAGE ITEM NAME : ASU     (FUNCTION : A)
                    UPPER LIMIT : 8.00
                    LOWER LIMIT : 0.00
                  DEFAULT VALUE : 0
          DATA OFFSET IN BUFFER :   55

QUESTION LABEL :  RUN HOURS?
------------------------------

    - ANSWER DEFINITION :
                          INPUT : KEYBOARD
                        LIGHT # : 14
                           TYPE : REAL
                IMAGE ITEM NAME : ARUN    (FUNCTION : A)
                    UPPER LIMIT : 8.00
                    LOWER LIMIT : 0.00
                  DEFAULT VALUE : -1
          DATA OFFSET IN BUFFER :   57

* LENGTH OF STORAGE FOR A U QUESTIONS SEQUENCE :   58
```

```
INFORMATION ADDED BY THE SYSTEM :
*********************************

         - TRANSACTION ID. :
          DATA OFFSET IN BUFFER :    1
                IMAGE ITEM NAME : TRID    (FUNCTION : A)

         - TERMINAL # :
          DATA OFFSET IN BUFFER :    3
                IMAGE ITEM NAME : TMNO    (FUNCTION : A)

         - DATE :
          DATA OFFSET IN BUFFER :    4
                IMAGE ITEM NAME : TRDTE   (FUNCTION : A)

         - TIME OF DAY :
          DATA OFFSET IN BUFFER :    7
                IMAGE ITEM NAME : TRTIM   (FUNCTION : A)

DATA COLLECTED STORAGE :
************************

                      FILE NAME : DATAFL
                           CR # : 24

  * TRANSACTION SPECIFICATION LENGTH :  267 WORDS
```

MINIMIZING SYNONYMS IN AN IMAGE/1000 DATA BASE

Audrey Dickey/HP Data Systems Division

Minimizing synonyms in an IMAGE data base is an effective method of improving data base performance. There are three ways that the user has at his disposal to decrease synonyms. These methods range from the very simple, choosing the proper capacity, to more difficult, devising a new hash algorithm. Use of any or all of these methods can result in decreased access time for data base users.

Before discussing how to minimize synonyms, a definition of synonyms is in order. Entries are placed into master data sets based on the value returned after hashing the key value. The result of this hashing is a relative record number which determines where in the data set the entry is placed. If more than one entry hashes to the same relative record number, synonyms are created. The first entry to hash to a given location is called the primary entry and subsequent entries hashing to the same record location are called synonyms. The primary entry resides at the proper relative record number and synonyms of that primary entry reside in empty records but are connected via links to the primary. When storing a synonym, IMAGE locates the primary entry by hashing, determines that the new entry is a synonym, finds a free record to store the synonym and searches to the end of the synonym chain in order to add the new entry to that chain. If the space occupied by a synonym is needed for a primary entry, the synonym must be moved to another free record space and links must be adjusted accordingly. When attempting to retrieve a synonym, IMAGE again must first locate the primary entry by hashing and then search the chain to locate the desired entry.

In an ideal data base, there would be no synonyms. Synonyms take more time than primary entries to store, maintain and locate. They have a direct effect on data base performance and so a data base should be designed to minimize synonyms as much as possible. There are three factors under the control of the data base designer which can affect the number of synonyms in a data base and therefore affect data base performance. They are 1) the number chosen as data set capacity, 2) the amount of free or empty records in a data set and 3) the hashing algorithm. Note that because entries are placed in detail data sets strictly in a first-in order, any discussion of synonyms applies only to master data sets.

The result of the hashing algorithm is a positive integer which is returned to the IMAGE subroutines. The IMAGE subroutines calculate the actual relative record number by the following formula:

```
R=(H mod C) + 1
```

where R is the relative record number, H is the result of the hash, and C is the capacity of the data set. A better distribution will be arrived at if the capacity is a prime number. For example, in a sample data set of capacity 100, of the 75 records in the data set, 50 were synonym records. If the capacity is changed to 101, a prime number, one record was a synonym. Because making the capacity a prime number is an easy thing to do with no adverse effects, it is strongly recommended that the capacities of all master data sets be a prime number.

Another factor under the control of the data base designer is the number of free records in the data set. The more empty space in a data set, the less likely a synonym will occur. Although it is difficult to set a given number as the ideal percentage of free records in a data set, a guideline is that at least twenty per cent of the data set should be empty. Once a data set is 80% or more full, performance degradation increases. A data set that is only 70% full is good — 60% full would be even better. In a sample data set, with 1100 records, a capacity of 1103 produced 402 synonyms. Increasing the capacity to 1373 so that the data set is approximately 80% full gives us 381 synonyms. A capacity of 1831 (60% full) produces only 266 synonyms.

It is not always possible to have a large number of empty records in a data set because of the limitation of 32767 records per data set or because there is physically not enough disc space in the system. It is the capacity, rather than the actual number of records in a set, that determines the size of the file. If the projected number of records in the data set is 32000, the capacity cannot be set to allow twenty or thirty per cent free records. It is then time to look at the third alternative, which is to change the hashing algorithm.

The hashing algorithm currently used by IMAGE consists of summing the first eight words of the key, with a right shift one bit of the sum to insure that the result is positive. This algorithm produces a positive one-word integer which is the H in the formula shown above. The results of this algorithm provide better distribution for certain types of keys than for others. If the key is an integer, the distribution is relatively even. For a character key greater than sixteen characters, the distribution could be very poor. The poorest distribution comes from what is referred to as an information poor key. This is a key with very little variation in value from entry to entry. For example, consider a part numbering scheme that contains four characters for the division, four characters for the department, four characters for the part type, and four characters for a sequential number. Although the key is sixteen characters long, unique information for any given part type is contained only in four of those characters. The key is poor in unique information. If the unique information is not contained in the first sixteen characters, the situation is even worse, since the hashing algorithm only looks at the first eight words of the key. There are two solutions to an information poor key — change the key or change the hash algorithm. Changing the key to increase the unique information may not be possible or practical. For example, although the key numbering scheme described above may be poor for IMAGE purposes, it may be very good for material control uses. By changing the hash algorithm, the unique information can be isolated to maximize even distribution. As an example, a data set was built containing an information poor key. Each key contained sixteen characters, fourteen of which were identical in every entry and two characters which were each one of fifty-nine printable characters. Using the standard hashing algorithm with 1100 records and a capacity of 1373, there were 381 synonyms produced. Substituting an algorithm that ignored the first seven words of the key, the number of synonyms went down to 99. If it is known that a key follows a certain pattern, a customized algorithm can be written that will extract the maximum amount of information from the key values.

In order to substitute a customized hashing routine, a function named HASH must be written that handles two parameters — starting word of the key and key length in words. The function should return a positive integer in the A-register. If this routine is relocated with the user program, it will be substituted for the standard library routine by the same name. Remember that if you are using a non-standard hash routine, to reload all IMAGE programs, including QUERY, with the new routine. Figure A contains an example of a function HASH which could be substituted for the standard HASH routine. Because substituting your own hash routine does modify the standard software, Hewlett-Packard does not support user-written routines or any problems that routine may cause. However, because of the modularity of the hash function, any impact that a user-written hash routine may have should be limited to the performance of the data base. It should be noted, however, that not consistently using the same hash routine every time the data set is accessed will affect data base integrity.

In order to evaluate whether you do have a problem with synonyms or whether a proposed hashing algorithm will improve distribution, a program can be written that will calculate the number of synonyms in a data set. The user can substitute various capacities and use the standard hash routine or substitute a new one. Figure B shows an example of such a program. This program reads an existing data set and calculates the number of synonyms as a result of various user-supplied capacity values and/or a user-supplied hash routine. By using such an analyzing program, several capacities and/or hash routines can be tried before actually rebuilding the data base.

If a user feels that data base performance is less than it could be, the following steps should be taken.

1. Use an analyzer program to determine if there are large numbers of synonyms in the data base.

2. Experiment with different values for capacity and see what effect they have on the number of synonyms. Use the analyzer to do this.

3. Examine the key to see if a better hashing algorithm can be devised.

4. Optimize the new hash algorithm to minimize the number of synonyms, again using the analyzer.

5. Rebuild the data base using the chosen capacity value and, optionally, the new hash routine. If changing the hash routine, remember to reload QUERY, DBBLD and all user program's that access the data base.

Following these steps allows the user to intelligently restructure his data base to improve performance with a minimum of effort on his part.

I wish to thank John Koskinen for the use of his HASH routine.

```
PAGE 0001  FTN.   6:08 PM  WED., 29  NOV., 1978

      FTN4,L
C
C          THIS FUNCTION RETURNS A PSEUDO RANDOM NUMBER BETWEEN
C          THE VALUES OF ZERO AND 32767 WHICH CAN BE USED AS A
C          HASHED RECORD NUMBER IN AN IMAGE DATA BASE APPLICATION.
C
      INTEGER FUNCTION HASH(IARG,LEN)
C
C          IARG = BEGINNING WORD OF THE KEY
C          LEN = LENGTH IN WORDS OF IARG
C
      DIMENSION IARG(8),IDUM(12),IRET(3)
      DOUBLE PRECISION D(4),DF
      EQUIVALENCE (IDUM,D),(DF,IRET)
C
C          INITIALIZE HASH VALUE AND DOUBLE PRECISION NUMBERS:
C          HASH=0, D (MANTISSA = .1, EXPONENT = 0)
C
      HASH=0
      D(1)=.1D0
      D(2)=.1D0
      D(3)=.1D0
      D(4)=.1D0
      IF(LEN.GT.8)LEN=8
C
C          BRANCH TO LENGTH OF KEY IN WORDS
C          LOADING MANTISSA OF D WITH TWO BYTE VALUES
C          SPREAD KEY ACROSS THE FOUR DOUBLE PRECISION
C          NUMBERS
C
      GO TO (10,20,30,40,50,60,70,80),LEN
80    IDUM(11)=IARG(8)
70    IDUM(08)=IARG(7)
60    IDUM(05)=IARG(6)
50    IDUM(02)=IARG(5)
40    IDUM(10)=IARG(4)
30    IDUM(07)=IARG(3)
20    IDUM(04)=IARG(2)
10    IDUM(01)=IARG(1)
C
C          MULTIPLY AND PLACE RESULT IN RETURN VARIABLE
C
      DF=D(1)*D(2)*D(3)*D(4)
C
C          EXTRACT THE MIDDLE PRECISION WORD OF
C          CALCULATED VALUE, USE ONLY 15 DATA BITS,
C          AND NOT THE SIGN BIT
C
      HASH=IAND(IRET(2),77777B)
      RETURN
      END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00207      COMMON = 00000
```

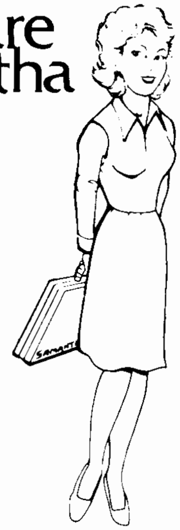
```
PAGE 0001  FTN.   4:05 PM  THU., 30  NOV., 1978

FTN4,L
C
C             THIS PROGRAM TAKES AN EXISTING DATA BASE AND
C             CALCULATES THE NUMBER OF SYNONYMS PRODUCED
C             WITH THE USER HAVING THE CAPABILITY TO SUPPLY
C             VARIOUS VALUES FOR CAPACITY.  IT IS ALSO POSSIBLE
C             TO SUBSTITUTE A USER HASH ROUTINE FOR THE STANDARD HASH
C             ROUTINE.
C
      PROGRAM HASHR(3,80)
      DIMENSION ILU(5),ILEVL(3),INAME(3),ILIST(4),IBUF(10),ID(3)
      DIMENSION IDCB(2576),ISIZE(2),ISTCH(3),ISTRNG(128),ISTAT(4)
      DIMENSION IBFR(256)
      INTEGER HASH
      DATA ILIST/1,2HHA,2HSH,2HR /
      DATA ISTCH/2H&H,2H&H,2H&H/
      DATA ISIZE/256,128/
C
C             THERE ARE TWO SCHEDULING PARAMETERS - LOG LU
C             AND LIST LU.  THE SYNONYM MAP AND THE STATISTICS
C             ARE OUTPUT TO THE LIST LU AND THE STATISTICS
C             ARE OUTPUT TO THE LOG LU.
C
      CALL RMPAR(ILU)
      IL=ILU(1)
      ILST=ILU(2)
C
C             THE USER SUPPLIES THE DATA BASE NAME, SECURITY
C             CODE, LEVEL WORD, A CARTRIDGE NUMBER FOR A SCRATCH
C             FILE AND THE NAME OF THE KEY.
C
      WRITE(IL,100)
100   FORMAT("ENTER DATA BASE NAME")
      READ(IL,200)(INAME(I),I=1,3)
200   FORMAT(3A2)
      WRITE(IL,300)
300   FORMAT("ENTER SECURITY CODE")
      READ(IL,*)ISEC
      WRITE(IL,500)
500   FORMAT("ENTER LEVEL WORD")
      READ(IL,600)(ILEVL(I),I=1,3)
600   FORMAT(3A2)
      WRITE(IL,700)
700   FORMAT("ENTER CARTRIDGE NUMBER")
      READ(IL,*)ICR
      WRITE(IL,800)
800   FORMAT("ENTER NAME OF ITEM TO BE HASHED")
      READ(IL,900)(ID(I),I=1,3)
900   FORMAT(3A2)
C
C             $H$H$H IS A SCRATCH FILE WHICH CONTAINS THE SYNONYM MAP
C
C             THIS IS THE INITITALIZATION SECTION, ENTERED ONLY ONCE.
C             THE SCRATCH FILE IS CREATED (FIRST IT'S PURGED JUST
C             IN CASE IT'S STILL HANGING AROUND), THE DATA BASE IS
```

```
PAGE 0002  HASHR  4:05 PM  THU., 30  NOV., 1978

C           OPENED AND DBINF IS CALLED TO GET THE POSITION OF THE
C           KEY.
C
      CALL PURGE(IDCB,IERR,ISTCH,0,ICR)
      CALL CREAT(IDCB,IERR,ISTCH,ISIZE,2,0,ICR,2560)
      IF(IERR.LT.0)GO TO 8100
      CALL DBINT(INAME,ISEC,ILIST,ISTAT)
      IF(ISTAT(1).NE.0)GO TO 8000
      CALL DBOPN(INAME,ILEVL,ISEC,1,ISTAT)
      IF(ISTAT(1).NE.0)GO TO 8020
      ITYPE=2HI
      CALL DBINF(ITYPE,5,ID,IBUF)
      IF(ISTAT(1).NE.0)GO TO 8040
      INUM=IBUF(2)
      CALL DBINF(ITYPE,2,INUM,IBUF)
      IF(ISTAT(1).NE.0)GO TO 8040
      IDSET=IBUF(9)
      IOFF=IBUF(8)
      ILENG=IBUF(7)
C
C           THIS SECTION IS ENTERED ONCE FOR EACH ITERATION.  IT
C           ZEROES OUT THE SYNONYM MAP IN THE SCRATCH FILE, SETS
C           THE ENTRY COUNT TO ZERO AND RESETS THE RELATIVE
C           RECORD POINTER TO THE BEGINNING OF THE FILE.  NOW
C           THE PROGRAM IS READY TO ACCEPT A TEST CAPACITY VALUE
C           FROM THE USER.
C
 1200 DO 1000 I=1,128
      ISTRNG(I)=0
 1000 CONTINUE
      DO 1100 I=1,256
      CALL WRITF(IDCB,IERR,ISTRNG,128,I)
      IF(IERR.NE.0)GO TO 8120
 1100 CONTINUE
      ICNT=0
      CALL DBGET(IDSET,3,ISTAT,IBFR,0)
      IF(ISTAT(1).NE.0)GO TO 8300
      WRITE(IL,1300)
 1300 FORMAT("ENTER TEST CAPACITY")
      READ(IL,*)ICAP
      IF(ICAP.LE.0)GO TO 9000
C
C           THE EXISTING DATA SET IS READ SERIALLY AND THE HASH
C           VALUE CALCULATED FOR EACH KEY VALUE.  NOTE THAT THE HASH
C           ROUTINE SUPPLIED BY THE USER MUST BE REFERENCED BY SOME
C           OTHER NAME THAN "HASH" WHEN RUNNING THE ANALYZER SINCE THE
C           STANDARD HASH IS USED BY THE IMAGE SUBROUTINES TO
C           RETRIEVE ENTRIES FROM THE DATA BASE.
C
C           AFTER THE RELATIVE RECORD NUMBER IS CALCULATED IT IS USED TO
C           INCREMENT A COUNTER IN THE SYNONYM MAP.  ALSO THE ENTRY
C           COUNT IS INCREMENTED.
C
 2000 CONTINUE
      CALL DBGET(IDSET,2,ISTAT,IBFR,IARG)
```

```
PAGE 0003  HASHR  4:05 PM  THU., 30  NOV., 1978

      IF(ISTAT(2).EQ.0)GO TO 7000
      IF(ISTAT(1).NE.0)GO TO 8200
      IREG=MHASH(IBFR(IOFF),ILENG)
      IMOD=MOD(IREG,ICAP)+1
      IREC=IMOD/128
      IWORD=IMOD-(IREC*128)+1
      IREC=IREC+1
      CALL READF(IDCB,IERR,ISTRNG,128,IREAD,IREC)
      ISTRNG(IWORD)=ISTRNG(IWORD)+1
      ICNT=ICNT+1
      CALL WRITF(IDCB,IERR,ISTRNG,128,IREC)
      GO TO 2000
C
C             ONCE THE DATA SET HAS BEEN READ AND THE ENTRY COUNT
C             IS DETERMINED, THE STATISTICS CAN BE CALCULATED.  IF
C             THE TEST CAPACITY IS LESS THAN THE NUMBER OF RECORDS,
C             NO RESULTS CAN BE CALCULATED.
C
 7000 CONTINUE
      IF(ICNT.LE.ICAP)GO TO 7020
      WRITE(IL,7010)
 7010 FORMAT("TEST CAPACITY IS LESS THAN NUMBER OF RECORDS IN DATA SET."
     1/" NO RESULTS WILL BE PRINTED.")
      GO TO 1200
 7020 CONTINUE
      IUN=0
      ISYN=0
      ITO=0
      IUSED=ICAP/128+1
      ICOUNT=ICAP
      DO 7100 I=1,IUSED
      CALL READF(IDCB,IERR,ISTRNG,128,IREAD,I)
      DO 7050 J=1,128
      ICOUNT=ICOUNT-1
      IF(ICOUNT.LT.0)GO TO 7060
      IF(ISTRNG(J).NE.0)GO TO 7030
      IUN=IUN+1
      GO TO 7050
 7030 CONTINUE
      ITO=ITO+1
      IF(ISTRNG(J).EQ.1)GO TO 7050
      ISYN=ISYN+ISTRNG(J)-1
 7050 CONTINUE
 7060 CONTINUE
      WRITE(ILST,7075)(ISTRNG(L),L=1,128)
 7075 FORMAT(16(8(I5,2X)/))
 7100 CONTINUE
      WRITE(ILST,7150)ICAP
 7150 FORMAT(" THE DATA BASE CAPACITY IS ",I5)
      WRITE(ILST,7175)ICNT
 7175 FORMAT(" THE NUMBER OF RECORDS IN THE DATA BASE IS ",I5)
      WRITE(ILST,7200)ISYN,ITO,IUN
 7200 FORMAT(" THE NUMBER OF SYNONYMS IS ",I5,/," THE NUMBER OF LOCATION
     1S HASHED TO IS ",I5,/," THE NUMBER OF LOCATIONS WITH NOTHING HASHE
     2D TO IS ",I5)
```

```
PAGE 0004  HASHR   4:05 PM  THU., 30  NOV., 1978

            SYN=ISYN
            CNT=ICNT
            PRCNT=(SYN/CNT)*100.
            WRITE(ILST,7300)PRCNT
      7300  FORMAT(" PERCENTAGE OF SYNONYMS IS ",F5.2,"%")
            IF(ILST.EQ.6)WRITE(ILST,7350)
      7350  FORMAT("1")
            IF(ILST.EQ.IL)GO TO 7400
            WRITE(IL,7150)ICAP
            WRITE(IL,7175)ICNT
            WRITE(IL,7200)ISYN,ITO,IUN
            WRITE(IL,7300)PRCNT
      7400  CONTINUE
            GO TO 1200
      8000  WRITE(IL,8010)ISTAT(1)
      8010  FORMAT(" IMAGE ERROR NO ",I5," ON DB INTIALIZATION")
            GO TO 9500
      8020  WRITE(IL,8030)ISTAT(1)
      8030  FORMAT(" IMAGE ERROR NO ",I5," ON DB OPEN")
            GO TO 9500
      8040  WRITE(IL,8050)ISTAT(1)
      8050  FORMAT(" IMAGE ERROR NO ",I5," ON DB INFO")
            GO TO 9000
      8100  WRITE(IL,8110)IERR
      8110  FORMAT(" FMP ERROR NO ",I5," ON SCRATCH FILE CREATION")
            GO TO 9000
      8120  WRITE(IL,8130)IERR
      8130  FORMAT(" FMP ERROR NO ",I5," ON SCRATCH FILE INITIALIZATION")
            GO TO 9000
      8200  WRITE(IL,8210)ISTAT(1)
      8210  FORMAT(" IMAGE ERROR NO ",I5," ON DB GET")
            GO TO 9000
      8300  WRITE(IL,8310)ISTAT(1)
      8310  FORMAT(" IMAGE ERROR NO ",I5," ON DB GET RESET")
      9000  CONTINUE
            CALL DBCLS(0,ISTAT)
            IF(ISTAT.EQ.0)GO TO 9500
      9200  WRITE(IL,9300)ISTAT(1)
      9300  FORMAT(" IMAGE ERROR NO ", I5, " ON DB CLOSE")
      9500  CONTINUE
            CALL PURGE(IDCB,IERR,ISTCH,0,ICR)
            END

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 04209      COMMON = 00000
```

Software Samantha



Software Samantha
HP-1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road, Cupertino, California 95014

Software Samantha
HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road, Cupertino, California 95014

"Dear Samantha:

I have an RTE III System with 7905 disc. The system uses over 130 tracks and I need quite a few type 6 files. Therefore I have a lack of scratch tracks for swapping. I do not have disc space to be able to have an auxiliary disc (LU 3). On a newly generated system I do not have any problem. But it is necessary to make frequent changes in the existing programs and most of the time the programs become larger than before. Apparently, after a while I do not have enough contiguous available tracks which can be used for swapping.

Is there a utillity which can pack the system tracks, or is there an efficient way to replace disc resident programs without creating "holes" on the disc? On this system I have at least 15-20 programs non-dormant at a time, some of them are pretty big. Whenever there is not any existing utility, I would appreciate if you could give me some guidelines to write my own.

Very truly yours,

Ranjana Shah, Senior Programmer
Purvis Systems Incorporated
6901 Jericho Turnpike
Syosset, New York 11791"

Dear Ranjana,

You have brought out a very good point in your letter, and our lab people are considering solutions. However no such utility to pack the system tracks exists today. User implementation of such a program would be difficult and possibly hazaradous to your operating system. The following suggestions should help alleviate your problem.

First of all when generating your system, relocate as few programs as possible; idealy only the File Manager (FMGR), the loader (LOADR) and any memory resident programs you might have, need to be relocated at generation time. This will decrease your system size, making room for more type 6 files on LU 2. Then load all of your programs on-line and using SP,<program> to save them in type 6 files. In your WELCOM file you should use RP,<program> to restore these programs after boot-up.

If disc space is still a problem remember that programs do not have to be "SP'd" onto LU 2, they can go to any FMP cartridge (SP,<program>::cr). However they must be "RP'd" or run from LU 2 (or 3). The following sequence of File Manager calls from a transfer file or the WELCOM file can "RP" a program from any FMP cartridge:

```
:ST,<program>::<cr>,<program>::-2
:RP,<program>
:PU,<program>::-2
```

This sequence repeated for each program could resore all your programs at boot-up using a minimum amount of space on LU 2.

"Dear Software Samantha,

The DOS/RTE Relocatable Library Reference Manual states that "CLRIO is a dummy compatibility routine for use by the FORTRAN compilers (was used by BCS system)." Why does my RTE FORTRAN compiler generate a call to CLRIO in every FORTRAN program?

Can you explain what is meant by the phrase "does blocked input/output" in the description of MAGTP?

Sincerely,

Richard B. Gilbert
Gas Dynamics Laboratory
James Forrestal Campus
Princeton, New Jersey 80540"

Dear Richard,

Those are two very valid questions; I'm glad you asked them.

CLRIO is called by the Fortran IV compiler for reasons of "backward compatibility." As the manual notes, CLRIO was used by the BCS system. The call to CLRIO by the RTE Fortran compiler makes it possible to produce a binary relocatable module which will run on a BCS system.

Your question on MAGTP points out an error in the DOS/RTE Relocatable Library Reference Manual. According to my information sources, MAGTP does not perform blocked input/output. The manual will be corrected in the next edition.

Thanks for your interest.

Samantha invites all questions from our readers of a technical nature on any aspect of HP 1000 systems. All letters will be answered, whether or not they are chosen for inclusion in the Communicator.

Address: Software Samantha
HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road, Cupertino, California 95014

WORKING WITH MULTIPOINT

David R. Fullerton/HP Neely Santa Clara

So, you are at your desk with a lot of multipoint goodies from HP and you are wondering what to do.

Well, first check to see if you have everything.

Software:

91730-16001 %DVR07 — Multipoint Driver
91730-12001 %MPLIB — Multipoint Library
91730-16002 %EXMP — Multipoint Exercisor
91730-16003 %DSPMP — Multipoint Display Program
91730-16009 %AUTO7 — Multipoint Auto Restart Program

Manuals:

91730-90002 Multipoint Terminal Subsystem Users Guide
91730-90001 Multipoint Software Numbering Guide
12790-16009 Multipoint Terminal Interface Reference Manual

Hardware:

12790A Multipoint Interface Card and Cable (CPU end)
13232P Cable for the first terminal
13232Q or T Cables for the following terminals
additional 13234A 4K Memory Module for the terminal
13260C Asynchronous Multipoint Datacomm Interface or
13260D Synchronous Multipoint Datacomm Interface (both include interface card and firmware to upgrade the 2645 or 2658 terminal)

Note that there are many other possible configurations. Please reference the Multipoint Terminal Subsystem Users Guide for the part numbers for your particular configuration.

Now that all of these items are present, there are three steps to make multipoint work.

First, generate an RTE-IV or RTE-MIII system with the Multipoint software. This step is relatively straight forward, with examples in the Multipoint Users Guide.

Second, set the switches in the 2645/48 terminal and on the 12790A interface board. As the settings are described in several manuals, it can be confusing, so a list of settings that will get Multipoint up and running is enclosed. Install all the boards and cables.

Last, you need to do some control calls to initialize the Multipoint line and terminals. As the Multipoint software is very flexible, there are many possibilities. Enclosed is a sample program that will initialize the Multipoint line and one terminal and make it emulate a TTY.

Once this is up and running, the manuals contain a description of a lot of extra things you can do with multipoint. If you need any help with setting up applications or designing your multipoint system, remember that consulting services are available from the Systems Engineering department of your sales office.

HP 1000 Multipoint:

- Keyboard interface:

— all switches closed (normal operation)			
— except	D	open	Page Mode
	J	open	Places Terminator
	K	open	Clears Terminator caused by strap J
	T,U	closed	1/2 Datacomm Buffer.
— with modem	V	closed	if no continuous carrier.

- Asynchronous multipoint interface (13260C):

J17, J16	Closed	512 bytes buffer (recommended)
J15	Closed	Sense Curser Addressing
J14		
J13		Device ID
J12		0 = closed
J11		1 = open
J10		
J07	Closed	ASCII code
J06	Open	CRC-16 Parity check
J05	Closed	Sync characters not inserted
J04		
J03		Group ID
J02		0 = closed
J01		1 = open
J00		
INT	Open	
PL6 thru		
PL0	Open	
A4	Closed	Firmware module address
A9 thru		
A11	Open	
-12	Open	Normally open, closed if using 13232T power protect cable.
2SB	Closed	Selects 1 stop bit.

- Synchronous multipoint interface (13260D):

J17, J16	Closed	512 buffer (recommended)
J15	Closed	Sense Curser Addressing
J14 thru J10		Device ID (0 = closed, 1 = open)
J07	Closed	ASCII code
J06	Open	CRC-16 Parity check
J05	Closed	IBM 3270 mode disabled
J04 thru		Group ID
J00		(0 = closed, 1 = open)
-12	Open	Normally open, closed if using 13232T power protect cable.
A4	Closed	Firmware module address
A9 thru		
A11	Open	
RCLK	Open	Always open.
2400		Close one of the 3 switches to select speed if RCLK closed.
4800		
9600		

```
FTN4,L
      PROGRAM INIT
C
C     INITIALIZE THE LINE
C
      ICW=100000B+(20*256)
C     LLU=THE LINE LU
      CALL EXEC(3,2000B+LLU,ICW)
C
C     INITIALIZE THE TERMINAL
C
C     TLU=THE TERMINAL LU
C     GID=GROUP ID
C     DID=TERMINAL ID
      ICW=IAND(GID,77B)*64+IAND(DID,77B)
      CALL EXEC(3,2000B+TLU,ICW)
C
C     SET UP THE TERMINAL TO LOOK LIKE A TTY
C
      CALL EXEC(3,2300B+TLU,71401B)
      END
```

1979 JULIAN CALENDER

JANUARY 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	6 (6)
7 (7)	8 (8)	9 (9)	10 (10)	11 (11)	12 (12)	13 (13)
14 (14)	15 (15)	16 (16)	17 (17)	18 (18)	19 (19)	20 (20)
21 (21)	22 (22)	23 (23)	24 (24)	25 (25)	26 (26)	27 (27)
28 (28)	29 (29)	30 (30)	31 (31)			

FEBRUARY 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1 (32)	2 (33)	3 (34)
4 (35)	5 (36)	6 (37)	7 (38)	8 (39)	9 (40)	10 (41)
11 (42)	12 (43)	13 (44)	14 (45)	15 (46)	16 (47)	17 (48)
18 (49)	19 (50)	20 (51)	21 (52)	22 (53)	23 (54)	24 (55)
25 (56)	26 (57)	27 (58)	28 (59)			

MARCH 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1 (60)	2 (61)	3 (62)
4 (63)	5 (64)	6 (65)	7 (66)	8 (67)	9 (68)	10 (69)
11 (70)	12 (71)	13 (72)	14 (73)	15 (74)	16 (75)	17 (76)
18 (77)	19 (78)	20 (79)	21 (80)	22 (81)	23 (82)	24 (83)
25 (84)	26 (85)	27 (86)	28 (87)	29 (88)	30 (89)	31 (90)

APRIL 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1 (91)	2 (92)	3 (93)	4 (94)	5 (95)	6 (96)	7 (97)
8 (98)	9 (99)	10 (100)	11 (101)	12 (102)	13 (103)	14 (104)
15 (105)	16 (106)	17 (107)	18 (108)	19 (109)	20 (110)	21 (111)
22 (112)	23 (113)	24 (114)	25 (115)	26 (116)	27 (117)	28 (118)
29 (119)	30 (120)					

MAY 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1 (121)	2 (122)	3 (123)	4 (124)	5 (125)
6 (126)	7 (127)	8 (128)	9 (129)	10 (130)	11 (131)	12 (132)
13 (133)	14 (134)	15 (135)	16 (136)	17 (137)	18 (138)	19 (139)
20 (140)	21 (141)	22 (142)	23 (143)	24 (144)	25 (145)	26 (146)
27 (147)	28 (148)	29 (149)	30 (150)	31 (151)		

JUNE 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1 (152)	2 (153)
3 (154)	4 (155)	5 (156)	6 (157)	7 (158)	8 (159)	9 (160)
10 (161)	11 (162)	12 (163)	13 (164)	14 (165)	15 (166)	16 (167)
17 (168)	18 (169)	19 (170)	20 (171)	21 (172)	22 (173)	23 (174)
24 (175)	25 (176)	26 (177)	27 (178)	28 (179)	29 (180)	30 (181)

JULY 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1 (182)	2 (183)	3 (184)	4 (185)	5 (186)	6 (187)	7 (188)
8 (189)	9 (190)	10 (191)	11 (192)	12 (193)	13 (194)	14 (195)
15 (196)	16 (197)	17 (198)	18 (199)	19 (200)	20 (201)	21 (202)
22 (203)	23 (204)	24 (205)	25 (206)	26 (207)	27 (208)	28 (209)
29 (210)	30 (211)	31 (212)				

AUGUST 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
			1 (213)	2 (214)	3 (215)	4 (216)
5 (217)	6 (218)	7 (219)	8 (220)	9 (221)	10 (222)	11 (223)
12 (224)	13 (225)	14 (226)	15 (227)	16 (228)	17 (229)	18 (230)
19 (231)	20 (232)	21 (233)	22 (234)	23 (235)	24 (236)	25 (237)
26 (238)	27 (239)	28 (240)	29 (241)	30 (242)	31 (243)	

SEPTEMBER 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1 (244)
2 (245)	3 (246)	4 (247)	5 (248)	6 (249)	7 (250)	8 (251)
9 (252)	10 (253)	11 (254)	12 (255)	13 (256)	14 (257)	15 (258)
16 (259)	17 (260)	18 (261)	19 (262)	20 (263)	21 (264)	22 (265)
23 (266)	24 (267)	25 (268)	26 (269)	27 (270)	28 (271)	29 (272)
30 (273)						

OCTOBER 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
	1 (274)	2 (275)	3 (276)	4 (277)	5 (278)	6 (279)
7 (280)	8 (281)	9 (282)	10 (283)	11 (284)	12 (285)	13 (286)
14 (287)	15 (288)	16 (289)	17 (290)	18 (291)	19 (292)	20 (293)
21 (294)	22 (295)	23 (296)	24 (297)	25 (298)	26 (299)	27 (300)
28 (301)	29 (302)	30 (303)	31 (304)			

NOVEMBER 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1 (305)	2 (306)	3 (307)
4 (308)	5 (309)	6 (310)	7 (311)	8 (312)	9 (313)	10 (314)
11 (315)	12 (316)	13 (317)	14 (318)	15 (319)	16 (320)	17 (321)
18 (322)	19 (323)	20 (324)	21 (325)	22 (326)	23 (327)	24 (328)
25 (329)	26 (330)	27 (331)	28 (332)	29 (333)	30 (334)	

DECEMBER 1979

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1 (335)
2 (336)	3 (337)	4 (338)	5 (339)	6 (340)	7 (341)	8 (342)
9 (343)	10 (344)	11 (345)	12 (346)	13 (347)	14 (348)	15 (349)
16 (350)	17 (351)	18 (352)	19 (353)	20 (354)	21 (355)	22 (356)
23 (357)	24 (358)	25 (359)	26 (360)	27 (361)	28 (362)	29 (363)
30 (364)	31 (365)					

COMMUNICATOR/1000 INDEX

Editor

Now seems a good time to print an index of all COMMUNICATOR/1000 articles as we close out Volume 2 with this issue. Below are two cross references of all past issues in both Volume 1 and Volume 2. Remember that Volume 1 issues 1 through 12 were titled COMPUTER SYSTEM COMMUNICATOR and it was not until Volume 1 issue 13 that the COMMUNICATOR was split into the COMMUNICATOR/1000, the COMMUNICATOR/2000 and the COMMUNICATOR/3000.

At any rate, the first index lists all articles sorted by volume, issue and page number. The second index is sorted by category, title and issue. A key for the category abbreviations follows:

UQ	User's Queue
OS	Operating Systems
IN	INstrumentation
CO	COmputation
OM	Operations Management
HA	HArdware Notes
OC	OEM Corner
BI	BIt Bucket
BU	BUlletins

Hopefully you will find these indexes valuable in your work.

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY CATEGORY

COMMUNICATOR INDEX BY CATEGORY

CAT	VOLUME	ISSUE	TITLE	AUTHOR	PAGE
	2	5	Notes on RTE Timeout	Del Kittendorf	15
	2	5	Operating System Drivers	Carl Reynolds	9
	1	2	Power Fail Auto Restart in RTE-B	Joe Deisel	47
	1	2	RTE Batch Spool Monitor	Hal Sindler	45
	1	13	RTE Performance	Lyle Weiman	20
	1	2	RTE-B Memory Requirements	Joe Diesel	44
	1	4	RTE-II/III & 21MX FFP	Jim Bridges	176
	1	8	RTE-II/III Class Table Structure	Jim Bridges	367
	1	7	RTE-III & Partitioning of Memory	Jim Bridges	315
	2	4	RTE-IV - Getting It Together	Snow and Kapoor	6
	2	2	RTE-IV Memory Organization	David L. Snow	9
	2	6	Reclaiming Class Numbers	David Fullerton	5
	1	1	Reconfigure BCS for New Interfac	George Taylor	5
	1	14	Returning RTE-III Memory Size	Larry W. Smith	6
	1	15	SMUT	Larry W. Smith	9
	1	14	Segmentation	Lyle Weiman	11
	2	3	Spooling is Easy	Jim Bridges	24
	1	14	Swapping	Lyle Weiman	7
	1	1	The :ST,X Directive	Paul McGillicuddy	2
	1	8	The WHZAT Program	Sandy Martensen	368
	2	4	Type 6 Files	Harvey Bernard	11
	1	6	Using Sys Disc Space in RTE-2/3	Jim Bridges	259
	1	1	Writing DOS-3 Directions to File	R.K. Strand	2

COMMUNICATOR INDEX BY CATEGORY

COMMUNICATOR INDEX BY CATEGORY

COMMUNICATOR INDEX BY CATEGORY

COMMUNICATOR INDEX BY CATEGORY

CAT	VOLUME	ISSUE	TITLE	AUTHOR	PAGE
	1	12	Microprogramming Best-Sellers	Mark Beswetherick	589
	1	7	New Batch Spool Monitor Ref Manu	Peter Baker	329
	1	13	New Contributed Library Catalog		28
	1	14	New Contributed Programs	Melanie Van Vliet	35
	1	16	New Contributed Programs	Melanie Van Vliet	26
	1	17	New Contributed Programs	Melanie Van Vliet	13
	2	2	New Courses		60
	2	4	New Courses		40
	2	5	New Courses		32
	2	6	New Courses		86
	1	2	New Products for RTE Users	Dave Sanders	47
	1	13	New Release for LOCUS		28
	2	2	New Software Support Program	George Taylor	8
	1	10	New Training Course Data Sheet	Jane Seligson	484
	2	3	New Training Progam		53
	1	5	RTE Interactive Editor Manual	Carol Guddal	231
	1	12	RTE Microprogramming Software	Don Ried	588
	1	4	RTE-II With 21MX	Jim Bridges	184
	1	5	RTE-III A Guide for New Users	Joan Martin	231
	2	3	RTE-IV Upgrade Course		52
	2	4	RTE-IV Upgrade Course		38
	2	5	RTE-IV Upgrade Course		30
	2	4	Revised Courses		41
	2	4	Setting Up a Training Program		38
	2	6	Setting Up a Training Program		84
	2	5	Setting up a Training Program		30
	2	6	Software Sources for RTE-IV	John Koskinen	83
	1	13	Software Updates		32
	1	14	Software Updates		40
	1	15	Software Updates		23
	1	16	Software Updates		31
	1	17	Software Updates		22
	2	1	Software Updates		26
	2	2	Software Updates		51
	1	14	Training Courses		47
	1	15	Training Courses		30
	1	7	Training News Flash	Tom Lowe	328
	1	13	Training Schedule		42
	1	16	Training Schedule		39
	1	17	Training Schedule		30
	2	1	Training Schedule		34
	2	2	Training Schedule		59
	2	3	Training Schedule		55
	2	4	Training Schedule		42
	2	5	Training Schedule		34
	2	6	Training Schedule		88
	2	3	User Training Services		62

SOFTWARE SOURCES FOR RTE-IV

John Koskinen/HP Data Systems Division

The software sources for RTE-IV are available for your use. A software license is required and is available for a fee. The 92067X Software Sources Product is a set of computer source code used to construct an RTE-IV (92067A) Real Time Executive operating system and supporting subsystems, such as the File Manager, Loader, Assembler, FORTRAN compiler, and libraries. The software sources product is provided for customers who have a current HP Purchase Agreement that wish to modify or directly support portions of the RTE-IV operating system software. Purchasing the 92067X product gives the customer the right to use RTE-IV sources on one HP 1000 computer with minimum hardware as defined in the 92067A data sheet.

The sources product includes the following:

RTE-IV Operating System
RTE-IV System Library
Relocating Loader
RTE-IV System Generator
Switch Program
WHZAT Program
Log Track Table
RTE Assembler
Spool System
File Manager
Directory Manager
Batch Monitor Library
Editor
Utilities
RTE FORTRAN IV Compiler
Compiler Library
Cross-Reference Program
Multi-Terminal Monitor
Power Fail and Auto-Restart Routine
Configuration Extension
EMA Diagnostic
Flexible Disc Backup
Formatter Library
RTE/DOS Library
Decimal String Package
RTE Drivers Package

PREREQUISITES

The 92067Z product is available to customers who have a current HP Purchase Agreement and who have previously acquired the 92067A product separately or in an HP 1000 system. Purchase of the 92067X product requires the signing of a Software License Agreement and payment of the license fee listed in the Hewlett-Packard Corporate Price List. The License Agreement defines the appropriate use of the Software Sources and any derived object code.

The right to copy any derived binary code from the modified sources is also available as 92067Y and is similar to the 92067R product.

Contact your HP Sales Representative for further details.

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.

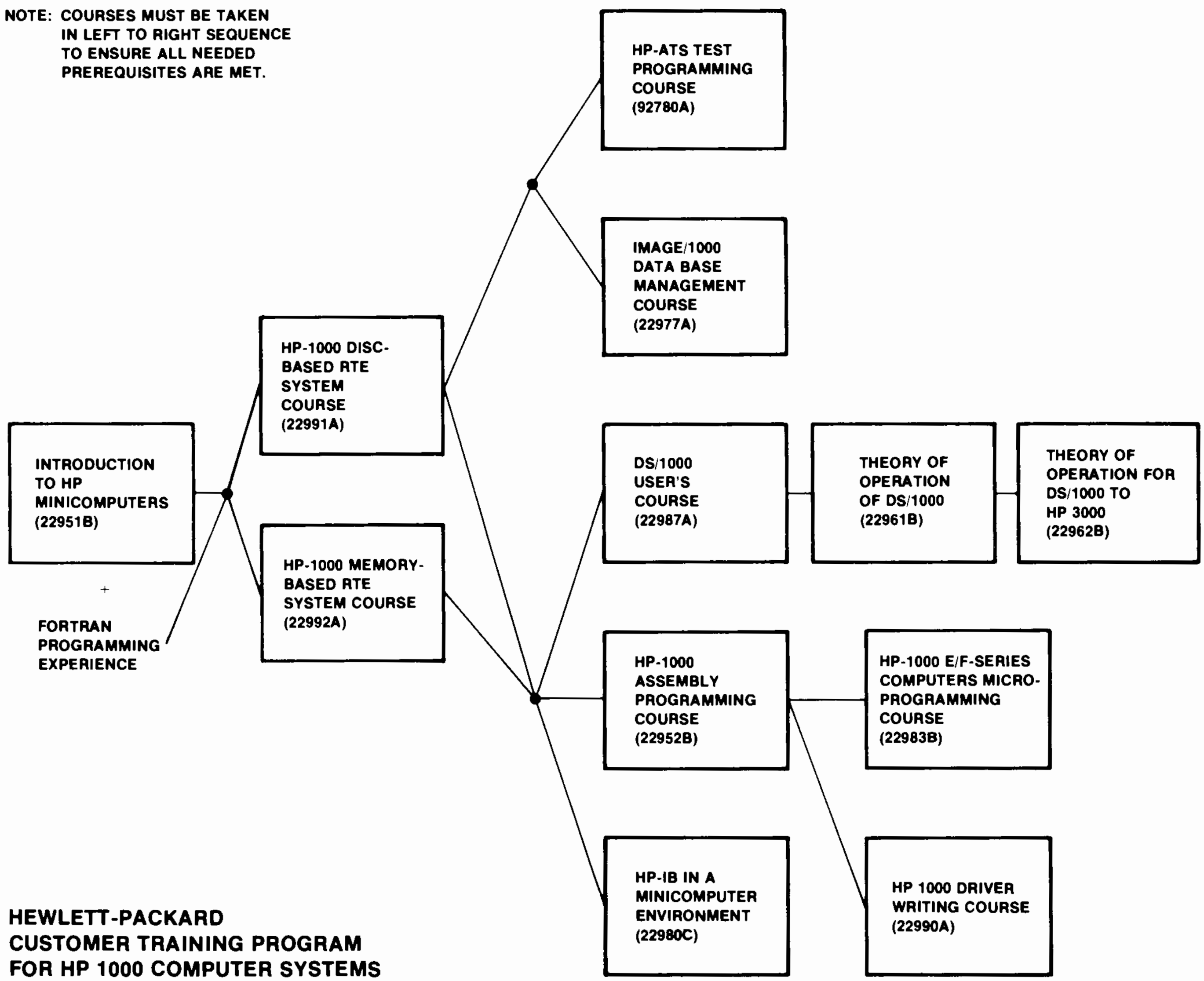
- Customize your computer for your application using the Microprogramming feature of the HP 1000. . .
  You should take the *HP 1000 E/F Series Computers Microprogramming* course (22983B).

- 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!



HEWLETT-PACKARD CUSTOMER TRAINING PROGRAM FOR HP 1000 COMPUTER SYSTEMS

NEW COURSES

In the last issue of the Communicator/1000, two new courses were added to the schedules that appear on the following pages. First is a brand new, two-week long **Memory-based RTE System Course (22992A)**, which covers the operation and programming of the RTE-M operating system. This course replaces the old one-week long RTE-M Course (22985A), which will shortly be obsoleted.

The second new course is the **Advanced RTE Workshop**, which is currently offered only at the Cupertino Customer Training Center. This course is taught by some of Hewlett-Packard's most experienced Systems Engineers and presents an in-depth discussion of the internal operation of the RTE-IV operating system.

More detailed information on both of these courses is given below.

22992A HP 1000 MEMORY-BASED RTE SYSTEM COURSE

Description: This course covers the use of the RTE-M operating system in an HP 1000 system environment. This includes program preparation using the standard flexible disc-based FORTRAN IV compiler, assembler, editor, relocating and absolute loaders; system software generation; and use of the file manager.

Length: 10 days.

Lab: Provides hands-on experience in operating programming and generating the RTE-M system, and in on-line program loading and removal.

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.

ADVANCED RTE WORKSHOP

Workshop Scope: This workshop will introduce the system analyst/programmer to the internal design and operation of the Real Time Executive operating system.

Who Should Attend: This 5-day workshop is designed for systems analyst/programmers who need to tune their systems for maximum performance. Since a considerable amount of material will be covered, attendees should be prepared for a very full week.

Prerequisites: The attendees must have at least 6 months experience in using the Real Time Executive and must have experience in using HP Assembly language.

Workshop Length: 5 days, 8: a.m. to 6:00 p.m.

Registration and Fee: Request for enrollment should be made through the Northern Neely training registrar (408) 996-9800, through your local HP Sales Office, or through your Sales Representative and should be accompanied by a purchase order or check for $800 made payable to Hewlett-Packard.

WORKSHOP SCHEDULE

OUTLINE FOR ADVANCED RTE IV WORKSHOP

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:00 9:00	Introduction Hardware Overview	Review Labs Operator Requests –Trace "ON,XYZ" From Keyboard to $XEQ	Resource Numbers LU Locks	Review Hw #3 Re-Entrant Processing –LIBR/LIBX REIO SAM –Users of SAM	Review Hw #2, 4 PowerII Systems Library Utilities
			COFFEE BREAK		
10:00 11:00 12:00	RTE Overview RTE Modules	Program Dispatching Partition Assignment	Review Hw #1 Program States –State Diagram –$LIST	–SAM Management I/O Drivers –Initialiszation –Continuation –Completion –Priviledged	Performance Measurement Lab Seminar
			LUNCH		
1:00 2:00	DMS –Phy./Log. Memory –RTE Maps Boot Process –Trace From Front Panel Thru $STRT	I/O Processing Overview Exec Calls –Trace Exec2 Call from MP Thru I/O	TBG Time Tick –Trace From Interr. to $XEQ		EMA –EMA in Fortran –EMA in Assembler –EMAST, MMAP –EMAP, EMIO
			COFFEE BREAK		
3:00 4:00 5:00 6:00	CMM4/DBUGR Lab	Completion Parity Errors Lab	Class I/O MTM –Trace From Keyboard Thru RPN Lab	Lab	Exam

TRAINING SCHEDULE

The current schedule for customer training courses on HP 1000 computer systems products is given in this section. Included are courses offered in the U.S., Europe, and in International areas 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 or International 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

To enroll in any of the courses listed in this publication please contact your local HP sales office and provide them with a class name, number and date you wish to attend along with a purchase order number from your company.

ADVANCED REGISTRATION

Hewlett-Packard training centers accept advanced registration for all courses. However, if a purchase order from your company has not been received at least two weeks prior to the start date of your class, reservation cannot be guaranteed.

ACKNOWLEDGEMENT

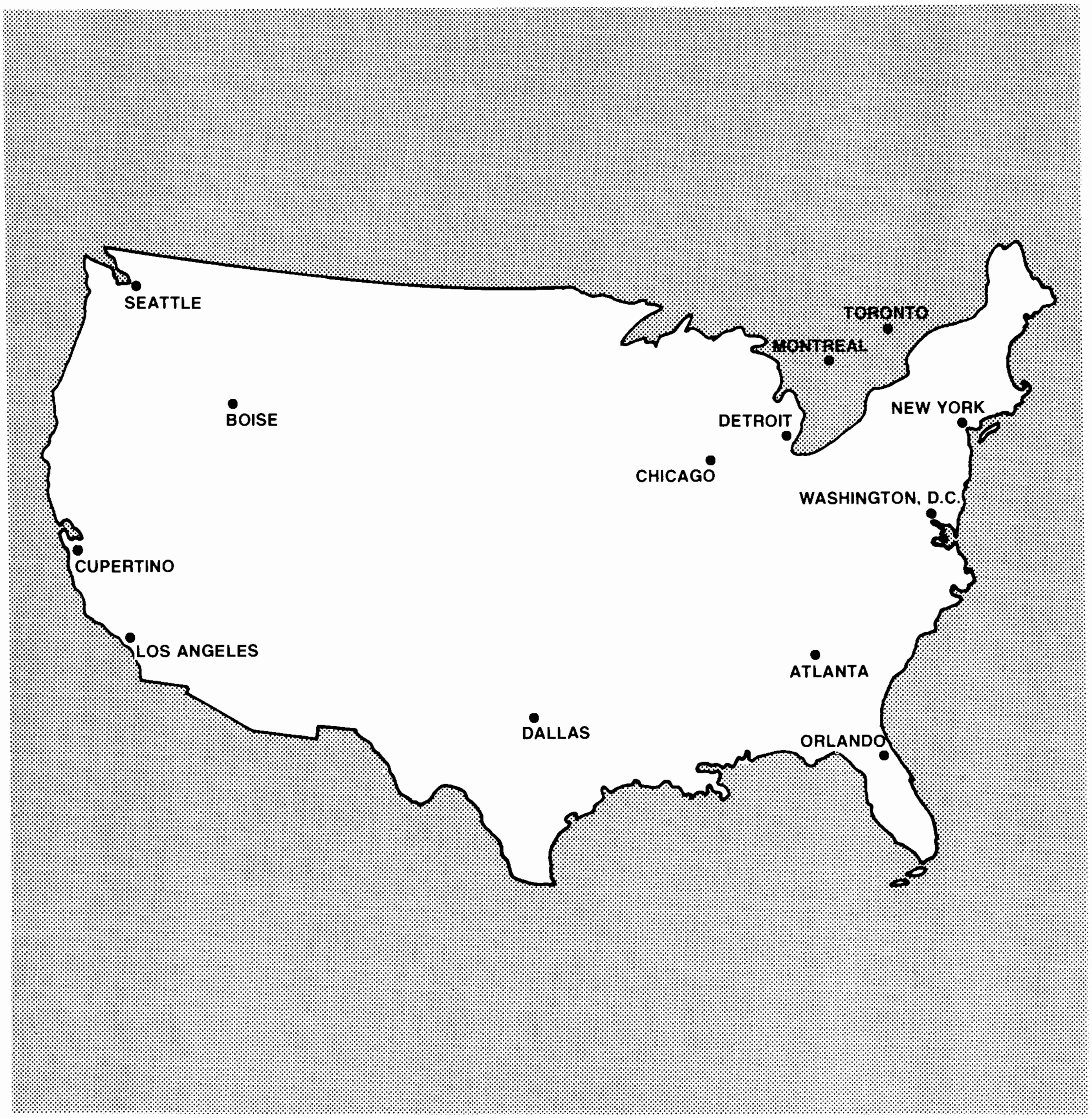
Within 10 working days of the receipt of your registration request, you will be sent a letter of confirmation and other local area information to help you plan your own hotel and travel accommodations.

CANCELLATION

Hewlett-Packard reserves the right to cancel any class due to insufficient enrollment. If this should occur, all enrollees will be notified as soon as possible in order to make other plans.

NORTH AMERICAN TRAINING CENTER LOCATIONS

The location of each North American training center is shown on the following page. Detailed addresses and phone numbers are given along with the schedule for each center.


SEATTLE
BOISE
CUPERTINO
LOS ANGELES
DALLAS
CHICAGO
DETROIT
TORONTO
MONTREAL
NEW YORK
WASHINGTON, D.C.
ATLANTA
ORLANDO

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

Course Number	Title	Length	Price	CUPERTINO Systems Engineering Center	LOS ANGELES Systems Engineering Center	WASHINGTON D. C. Systems Engineering Center	CHICAGO Systems Engineering Center	DALLAS (D) ATLANTA (A) Systems Engineering Centers	DETROIT Systems Engineering Center	NEW YORK Systems Engineering Center
22951B	Intro to HP mini's	4 days	400	Apr 2	Jan 8 Mar 12 Apr 30	Jan 8 Feb 12 Mar 19 Apr 16	Jan 29 Apr 2	May 21 (D)	Jan 15 Mar 26 May 14	Mar 5 May 7
22991A*	HP 1000 DISC RTE (Course includes RTE-IV operating system, batch spool monitor and file manager.)	10 days	1000	Jan 8 Jan 22 Feb 5 Feb 26 Mar. 12 Mar 26 Apr 16 Apr 30 May 14 Jun 4	Jan 15 Feb 5 Mar 19 Apr 16 Mar 7	Jan 15 Jan 29 Feb 26 Mar 12 Mar 26 Apr 16 Apr 30 May 14	Feb 12 Apr 16	Jan 29 (A) Mar 19 (D)		Mar 12 Apr 16 May 14
22992A*	HP 1000 Memory RTE	10 days	1000	Feb 26						
22977A*	IMAGE	5 days	500	Jan 22 Apr 23	Feb 26 May 21	Feb 5 Mar 12 May 21	Feb 26 Apr 30			Apr 2
22952B*	1000 ASMB	5 days	500	Feb 5 Apr 30	Jan 29 Apr 2 Jun 4	Jan 29 Mar 5 Apr 23	Mar 5			Mar 26 Apr 30
22987A*	DS/1000 User's Course	5 days	500	Mar 26		Jan 8 Apr 30				
22961B*	DS/1000 Theory of Op.	4 days	400			Jan 15 May 7				
22962B*	DS/1000 to HP 3000 Theory of Op.	1 day	100			Jan 19 May 11				
22990A*	RTE-Driver Writing	3 days	300	Feb 12 May 21		Feb 21 May 30				Apr 9

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

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

Course Number	Title		CUPERTINO Systems Engineering Center	LOS ANGELES Systems Engineering Center	WASHINGTON D. C. Systems Engineering Center	CHICAGO Systems Engineering Center	DALLAS (D) ATLANTA (A) Systems Engineering Centers	DETROIT Systems Engineering Center	NEW YORK Systems Engineering Center
	Length	Price							
22980C*	HP-IB Minicomputer Environment		Jan 15 Mar 19						
	4 days	400							
22983B*	HP 1000 E/F Microprogram-ming		Jan 29 May 7						May 14
	5 days	500							
	Advanced RTE Workshop		Jan 8 Mar 12 May 14						
	5 days	800							

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

**For Registration Information, call Cupertino Customer Training Center Registrar.

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

Course Number	Title / Length	Price	Data Systems Division (CUPERTINO)	Data Terminals Division (CUPERTINO)	Customer Service Division (CUPERTINO)	Boise Division (BOISE)	
92780A*	HP-AS Automatic Test System 5 days	1000	Feb 26				
13294A	Dev. Terminal 5 days	500		Jan 8 Feb 26			
22940A	2100 Maint. 10 days	1000			Jan 22 Feb 26 Mar 26 Apr 30		
91303A	HP 1000 Operational Maintenance 8 days	1000			Jan 9 Jan 29 Mar 6 Mar 27 Apr 17 May 15		
22942A	7900 Maint. 5 days	500			Jan 15 Feb 12 Apr 23		
91304A	HP Disc Drive Operational Maint. 5 days	500			Jan 8 Feb 5 Mar 12 Mar 19 Apr 16 May 14 May 21		
91302A	2645 Maint. 3 days	300					
22943A	7970B Maint. 5 days	600					
22944A	7970E Maint. 5 days	600					

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

U.S. TRAINING CENTER ADDRESSES

Atlanta

CUSTOMER TRAINING CENTER
450 Interstate North Parkway, NW
Atlanta, Georgia 30339
(404) 955-1500

Boise

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

Cupertino

CUSTOMER TRAINING CENTER
19320 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9800

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

Dallas

CUSTOMER TRAINING CENTER
201 E. Arapaho Road
Richardson, Texas
(214) 231-6101

Los Angeles

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

Washington, D.C.

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

New York

CUSTOMER TRAINING CENTER
120 Century Road
Paramus, N.J. 07652
(201) 265-5000

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title	Length	Boblingen	Amsterdam	Madrid	England Altrincham (A) Winnersh (W)	Milan (M) Rome (R)	Stockholm	Helsinki	Orsay	Vienna	Brussels
22951B	Intro to HP mini's	4 days 400		Feb 05 Apr 30 Aug 27		Jan 29 (A) Feb 26 (W) Mar 19 (A)		Jan 22 Apr 02 Oct 08			Sep 3	
22965B	RTE-II/III (Course includes RTE-II/III operating system, batch spool monitor and file manager.)	10 days								Jan 8 Feb 12 Mar 26 May 7 Jun 11	Jan 15	
22991A*	HP 1000 DISC RTE (Course includes RTE-IV operating system, batch spool monitor and file manager.)	10 days 1000		Jan 08 Feb 26 Apr 02 May 14 Jun 25 Sep 03 Oct. 15	Feb 26 Oct 29	Jan 08 (W) Feb 05 (A) Mar 04 (W) Mar 26 (A)		Jan 29 Mar 5 Apr 23 Sep 10 Oct 15 Nov 19	Jan 15 Mar 5		Mar 19 Sep 10	Jan 8 Mar 12 May 28 Oct 1
22985A	RTE-M	5 days								Mar 5		
22977A*	IMAGE	5 days		Apr 23 Jul 16 Oct 08	Mar 12 Nov 12	Jan 22 (W) Apr 23 (A)			Feb 5	Mar 12 Jun 25 Sep 24	Jan 29 Apr 02	
22952B*	1000 ASMB	5 days		Mar 19 Jun 11 Oct 01	Feb 19 Oct 22	Feb 19 (A)		Feb 26 Sep 03 Nov 12	Apr 02	Mar 5 May 08	Sep 24	
22987A*	DS/1000 User's Course	5 days		Feb 12 Sep 17		Mar 26 (W)			Feb 19	Feb 5		
22961B*	DS/1000 Theory of Op.	4 days		Sep 24		Apr 02 (A)						
22962B*	DS/1000 to HP 3000 Theory of Op.	1 day		Sep 28		Apr 6 (A)						
22990A*	RTE Driver Writing	3 days										
22980C*	HP-IB Minicomputer Environment	4 days		Apr 16 Aug 20						Apr 17		
22983B*	HP 1000 E/F Microprogramming	5 days										

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages 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							Feb 26			
22942A	7900 Maint. 5 days							Mar 5			
22945A	7905/06 Maint. 5 days							Feb 19 Apr 02			
22984A	7920 Maint. 5 days							Apr 09			
91302A	2645 Maint. 3 days							Jan 22			
22943A	7970B/E Maint. 5 days							Jan 15 Mar 26			
40270A	Intro to HP Computers 5 days							Jan 22 Apr 09 Jul 02			
22965B-H01	FORTRAN IV 5 days			Feb 12 Oct 15							

*These courses carry prerequisites. Refer to the training program diagram and discussion on the previous pages for more information.

EUROPEAN TRAINING CENTER ADDRESSES

Altrincham, England

Navigation Road
Altrincham
Cheshire WA14 1NU

Amsterdam, the Netherlands

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

Boblingen, Germany

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

Brussels, Belgium

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

Grenoble, France

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

Helsinki, Finland

Nahkahousuntie 5
00211 Helsinki 21
Tel: 90-692 30 31

Madrid, Spain

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

Milan, Italy

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

Orsay, France

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

Stockholm, Sweden

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

Vienna, Austria

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

Winnersh, England

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

INTERCONTINENTAL TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title / Length	CANADA Montreal	CANADA Toronto	AUSTRALIA Blackburn, VIC (B) Pymble, NSW (P)	JAPAN Tokyo (T) Osaka (O)	
22951B	Intro to HP mini's / 4 days	Feb 19**	Jan 08		Jan 15 (T) Feb 27 (T) Apr 09 (T) May 22 (T)	
22991A*	HP 1000 DISC RTE / 10 days / (Course includes RTE-IV operating system, batch spool monitor and file manager.)	Mar 19**	Feb 19	Feb 26 (B) May 21 (P) Jul 09 (B) Sep 17 (P) Oct 22 (B)	Jan 22 (T) Mar 05 (T,O) Apr 16 (T,O) May 28 (T) Jun 04 (O)	
22992A*	HP 1000 Memory RTE / 10 days				Feb 19 (T)	
22977A*	IMAGE / 5 days			Apr 30 (B) Jul 09 (P) Aug 20 (B) Nov 12 (P) Dec 03 (B)	Jun 25	
22952B*	1000 ASMB / 5 days			Jun 11 (P) Jul 30 (B) Oct 08 (P) Nov 12 (B)	Feb 05 (T) Mar 26 (T) Apr 16 (O) May 7 (T) Jun 11 (T)	
22987A*	DS/1000 User's Course / 5 days			May 28 (B) Oct 22 (P)	Jun 18 (T)	
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			Jun 18 (P) Aug 06 (B) Oct 01 (P) Nov 19 (B)	Apr 03 (T)	
22980C	HP-IB / 4 days				May 14 (T)	

*These courses carry prerequisites — refer to the training program diagram and discussion on the previous pages for more information.

**These courses are taught in French.

INTERCONTINENTAL TRAINING CENTER ADDRESSES

Blackburn, Australia

CUSTOMER TRAINING CENTER
31-41 Joseph Street
Blackburn, Victoria, Australia

Pymble, Australia

CUSTOMER TRAINING CENTER
31 Bridge Street
Pymble, New SouthWales, Australia

Montreal, Canada

CUSTOMER TRAINING CENTER
275 Hymus Boulevard
Pointe Claire, Quebec, Canada H9R1G7
(514) 697-4232

Toronto, Canada

CUSTOMER TRAINING CENTER
6877 Goreway Drive
Mississauga, Ontario, Canada, L4V 1M8
(416) 678-9430

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.

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

[X] **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

[X] **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 BPL 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 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.

Printed in U.S.A. 12/78

Part No. 5951-6111