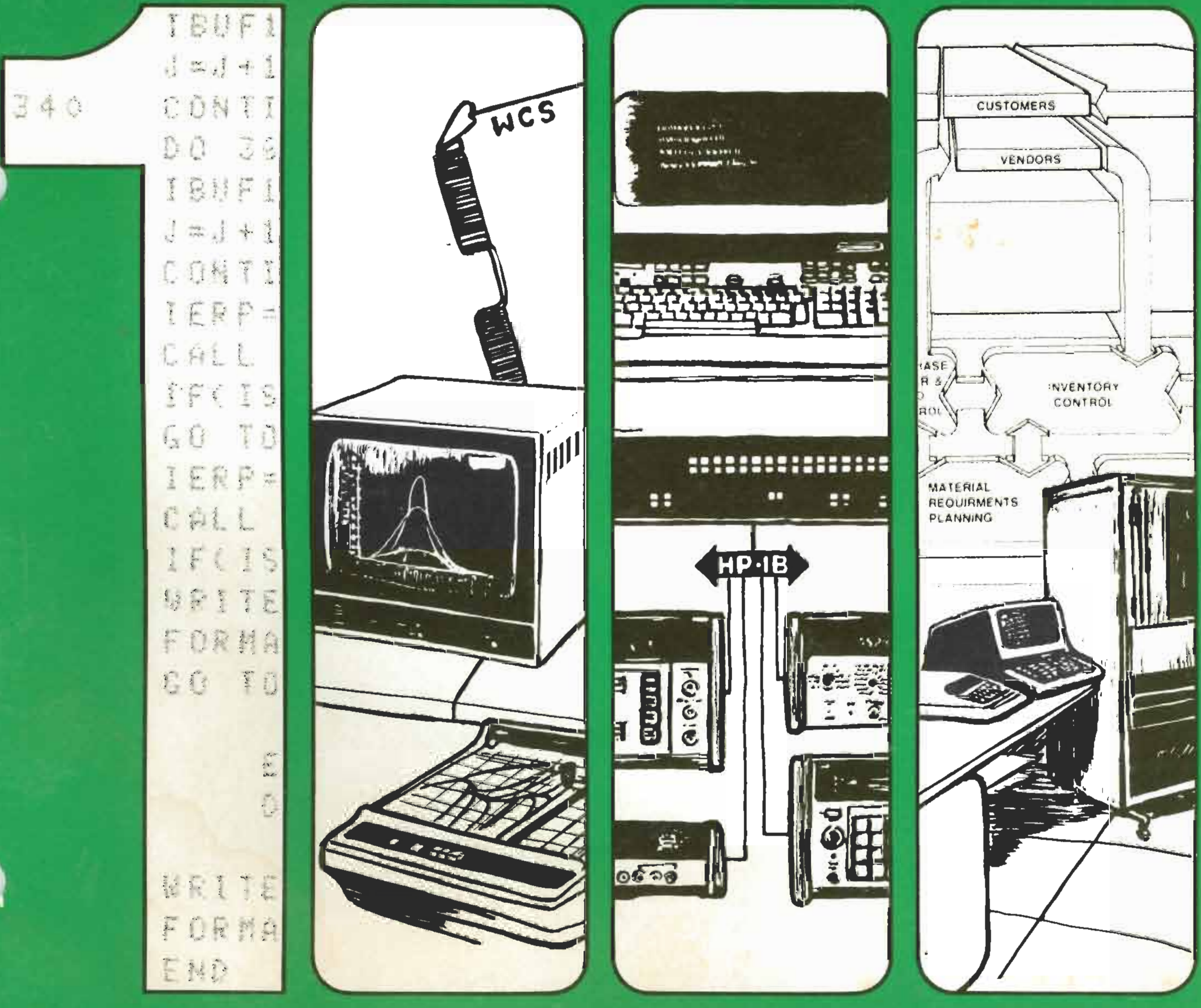
HEWLETT PACKARD

Hewlett-Packard Computer Systems COMMUNICATOR



1979

Volume III Issue 4

HEWLETT-PACKARD
COMPUTER SYSTEMS

Volume III
Issue 4

COMMUNICATOR/1000

Feature Articles

OPERATING SYSTEMS

17 **LOADING PROGRAM SEGMENTS FROM FMP FILES**
Larry W. Smith/HP Fullerton

28 **CONVERSION OF COMPUTER PROGRAMS TO HP 1000 RTE-IV**
Jack B. McAlister/Technology Development Corp.

INSTRUMENTATION

32 **THE FUNDAMENTALS OF HP-IB ADDRESSING**
Neal Kuhn/DSD Applications Development

OPERATIONS MANAGEMENT

38 **REMOTE DATA BASE ACCESS: HOW DOES IT WORK?**
Carol Jonas/DSD, Technical Marketing

Departments

EDITOR'S DESK

2 **ABOUT THIS ISSUE**
3 **BECOME A PUBLISHED AUTHOR IN THE COMMUNICATOR/1000**

USER'S QUEUE

5 **LOCUS ADDITIONS**
7 **LETTER TO THE EDITOR**

BIT BUCKET

8 **SOFTWARE SAMANTHA**
10 **DS/1000 TIME GETTER**
12 **IDCHK UTILITY PROGRAM**

BULLETINS

42 **INTRODUCING THE NEW IMAGE/1000 DBMS**
44 **ANNOUNCING DATACAP/1000 SOFTWARE PACKAGE FOR REAL-TIME DATA CAPTURE**
46 **HP SOFTWARE SUPPORT — FLEXIBLE AND USER ORIENTED**
48 **HP 1000 USERS ESTABLISH A USER GROUP**
50 **JOIN AN HP 1000 USER GROUP**
52 **HP 1000 SOFTWARE COURSES**
57 **INTERNATIONAL TRAINING SCHEDULES 79/80**

EDITOR'S DESK

ABOUT THIS ISSUE

This issue of the Communicator/1000 features two articles in the category OPERATING SYSTEMS and one each in OPERATIONS MANAGEMENT and INSTRUMENTATION. The two OPERATING SYSTEMS articles come from the field; one from an HP systems engineer and one from an HP customer. The other two articles were written by HP employees within Data Systems Division.

In the OPERATING SYSTEMS section, Larry W. Smith of HP's Fullerton, Ca. sales and service office contributes once again with an excellent article on a method he devised for overcoming the limit to the number of executable programs on an RTE system. His subroutine SLOAD allows programs to be stored on any disc LU, and executed without an ID segment being created for each program. Our customer contributor, Jack B. McAlister of Technology Development Corporation presents a step-by-step approach to converting Fortran programs from other operating systems to run on an RTE-IV system. Conversion of programs from one operating system to another can result in considerable time savings for customers who own several different systems, or who frequently interact with owners of other systems.

In the INSTRUMENTATION section, Neal Kuhn of HP's Data Systems Division writes an article which makes HP-IB addressing easy. His article will be extremely helpful to new HP-IB users who may be confused about the protocol for issuing different commands to various devices. Neal first explains how commands are structured, and then shows several methods of sending them to devices.

Lastly, in the OPERATIONS MANAGEMENT section, Carol Jonas of DSD's Technical Marketing Department submits an excellent article on remote access of IMAGE/1000 data bases. She outlines two separate ways to accomplish remote access; with QUERY and with programmatic RDBA calls.

As of this issue there has been a change in the separation of competition for calculators. HP field personnel will no longer be competing against HP divisions. HP division employees, including those in the Data Systems Division will be competing against each other. Of course, DSD's technical marketing department will still be ineligible for prizes.

Now, the moment you've all been waiting for! The winners of HP32E Calculators are:

Best Feature Article by a Customer	**CONVERSION OF COMPUTER PROGRAMS TO HP 1000 RTE-IV** Jack B. McAlister
Best Feature Article by an HP Division Employee not in Data Systems Technical Marketing	**THE FUNDAMENTALS OF HP-IB ADDRESSING** Neal Kuhn

Unfortunately, we had no competitors in the category of HP Field Employees since Larry Smith became ineligible after winning a calculator earlier this year. We hope to have other HP field personnel contributing to later issues, as well as continued correspondance from HP customers.

Hope you enjoy this issue of the Communicator/1000!

The Editor

BECOME A PUBLISHED AUTHOR IN THE COMMUNICATOR/1000 . . .

The COMMUNICATOR is a technical publication designed for HP 1000 computer users. Through technical articles, the direct answering of customers' technical questions, cataloging of contributed user programs, and publication of new product announcements and product training schedules, the COMMUNICATOR strives to help each reader utililize their HP 1000's more effectively.

The Feature Articles are clearly the most important part of the COMMUNICATOR. Feature Articles are intended to promote a significant cross-fertilization of ideas, to provide in-depth technical descriptions of application programs that could be useful to a wide range of users, and to increase user understanding of the most sophisticated capabilities designed into HP software. You might think of the COMMUNICATOR as a publication which can extend your awareness of HP 1000's to include that of thousands of users worldwide as well as that of many HP engineers in Data Systems factories at Cupertino, California and Grenoble, France.

To accomplish these goals, editors of the COMMUNICATOR actively seek technical articles from HP 1000 customers, HP Systems Engineers in the Field, and Marketing and R&D Engineers in the factories. Technical articles from customers are most highly valued because it is customers who are closest to real-world applications.

WIN AN HP-32E CALCULATOR!

Authoring a published article provides a uniquely satisfying and visible feeling of accomplishment. To provide a more tangible benefit, however, HP gives away three free HP-32E hand-held calculators to Feature Article authors in each COMMUNICATOR/1000 issue! Authors are divided into three categories. A calculator is awarded to the author of the best Feature Article in each of the author categories. The three author categories are:

1. HP 1000 Customers;

2. HP field employees;

3. HP division employees not in the Data Systems Division Technical Marketing Dept.

Each author category is judged separately. A calculator prize will be awarded even if there is only one entry in an author category.

Feature Articles are judged on the following bases: (1) quality of technical content; (2) level of interest to a wide spectrum of COMMUNICATOR/1000 readers; (3) thoroughness with which subject is covered; and, (4) clarity of presentation.

What is a Feature Article? A Feature Article meets the following criteria:

1. Its topic is of general technical interest to COMMUNICATOR/1000 readers;

2. The topic falls into one of the following categories —

   OPERATING SYSTEMS
   DATA COMMUNICATIONS
   INSTRUMENTATION
   COMPUTATION
   OPERATIONS MANAGEMENT

3. The article covers at least two pages of the COMMUNICATOR/1000, exclusive of listings and illustrations (i.e., at least 1650 words).

EDITOR'S DESK

There is a little fine print with regard to eligibility for receiving a calculator; it follows. No individual author will be awarded more than one calculator in a calendar year. In the case of multiple authors, the calculator will be awarded to the first listed author of the winning article. 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. Employees of Technical Marketing at HP's Data Systems Division factory in Cupertino are not eligible to win a calculator.

All winners of calculators will be announced in the issue of the COMMUNICATOR/1000 in which their articles appear. Again, all Feature Articles are judged by an impartial panel of three DSD Technical Marketing Engineers.

A SPECIAL DEAL IN THE OEM CORNER

When an HP 1000 OEM writes a Feature Article that is not only technically detailed and insightful but also application-oriented as opposed to theoretical, then that OEM may ask that the article be included in THE OEM CORNER. A Feature Article included in THE OEM CORNER may contain up to 150 words of pure product description as well as a picture or illustration of the OEM'S product or its unique contribution. HP's objective is twofold: (1) to promote awareness of the capabilities HP 1000 OEMs' products among all HP 1000 users; and, (2) to publish an article of technical interest and depth.

IF YOU'RE PRESSED FOR TIME . . .

If you are short of time, but still have that urge to express yourself technically, don't forget the COMMUNICATOR/1000 BIT BUCKET. It's the perfect place for a short description of a routine you've written or an insight you've had.

THE MECHANICS OF SUBMITTING AN ARTICLE

If at all possible please submit an RTE File containing the text of your article recorded on a Minicartridge (preferrably) or on a paper tape along with the line printer or typed copy of your article. This will help all of us to be more efficient. The Minicartridge will be returned to you promptly. Please include your address and phone number along with your article.

All articles are subject to editorship and minor revisions. The author will be contacted if there is any question of changing the information content. Articles requiring a major revision will be returned to the author with an explanatory note and suggestions for change. We hope not to return any articles at all; if we do, we would like to work closely with the author to improve the article. HP does, however, reserve the right to reject articles that are not technical or that are not of general interest to COMMUNICATOR/1000 readers.

Please submit your COMMUNICATOR/1000 article to the following address:

Editor, COMMUNICATOR/1000
Data Systems Division
Hewlett-Packard Company
11000 Wolfe Road
Cupertino, California 95014
USA

The Editor looks forward to an exciting year of articles in the COMMUNICATOR/1000.

With best regards,

The Editor

LOCUS ADDITIONS

The new contributed programs listed below are now available in Locus. Contact your local HP sales office to order Contributed Library programs.

22683-XXX47 P1640

P1640 is a program to expand the capabilities of the 1640 serial logic analyzer by providing:

- storage of data from the 1640 on memory,disk,tape,etc.
- alterable formatting of saved data for any type of code.
- formatted output to the line printer or a file.
- automatic initialization of the 1640 through transfer files.
- user doesn't need to know HP-IB but must have the hardware and software in the system.

The program is designed to be used interactively, but could be run by batch.

22683-10947	800 bpi mag tape	$50.00
22683-11947	1600 bpi mag tape	$50.00

22683-XXX48 DVM33 — RTE-M Disc Driver

DVM33 is a RTE-M disc driver which transfers data to and from expanded memory instead of the 9885 floppy disc drive. This provides the user with the capability of having an operating system completely memory resident, including program data files.

DVM33 supports standard read and write requests (EXEC 1 or 2) and is compatible with the existing RTE-M subsystems (FMGR, EDITM, ASMB, FTN4, etc).

Program INITL is used to initialize DVM33 after boot-up. It prompts the system console for the number of pages in physical memory and the number of pages in the operating system (MIII only).

Equipment, LU and interrupt table entries are the same as the floppy disc driver DVR33 exept DMA, buffering and time out should not be specified in the equipment table and the interrupt table requires only one (DUMMY) select code. Program INITL expects the disc to be LU2. DVM33 does not support equipment subchanges.

22683-13348	Minicartridge	$40.00

22683-XXX49 CAMAC DRIVER PACKAGE

The CAMAC driver package operates on a HP 2100 or HP 21MX computer under RTE-II, RTE-III, or RTE-IV.The package consists of

1. A CAMAC driver (I/O handler) (DVA54). This is a system resident routine allowing user program access to CAMAC compatible equipment.

2. An interactive diagnostic program (CAM2). This is an interactive program used for testing CAMAC equipment and the driver.

3. An array initialization subroutine (FILL). This subroutine will initialize an array to any desired value. All entries in the array will contain the same value. This subroutine is called by CAM2.

22683-13349	Minicartridge	$40.00

22683-XXX50 LTAPE--HP 1000 Mag Tape Dump Analyzer

This program allows a 9-track or 7-track magnetic tape of any density to be analyzed for such items as record lengths, parity errors, number of files, and tape controller status. The program performs a complete hardware and software check and prints resulting tape statistics as to total number of records and average record length.

22683-13350	Minicartridge	$40.00

LETTER TO THE EDITOR

Dear Sir,

With reference to the article FAST REAL TIME I/O UNDER RTE by John Pezzano in Communicator 1000 Vol. II, Issue no. 5, I noted that he states that DMA is not available when using $LIBR/LIBX calls to turn off the interrupt system. I would appreciate an expansion of why this is so, as I think that provided no DMA operation was currently in progress when $LIBR is called, then the DMA system is usable for I/O.

Yours sincerely,

K. Murdoch
for DIRECTOR:
NATIONAL ACCELERATOR CENTRE

Dear Sir,

You are entirely correct; the DMA system continues to be available after a $LIBR or $LIBX call is used to turn off the interrupt system. In fact, available DMA channels are denoted in the interrupt table by a zero entry.

Thanks for writing.

With best regards,

The Editor

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

Dear Samantha,

Those of us here at Helical have really appreciated the article entitled CREATING AND CLEARING EXTENTS from Volume III, Number 1. The program has been extremely helpful in keeping our disks organized and cleared out.

The program has prompted several questions from our staff, however. We would be grateful if you could question Alan Housley and any others on your staff to provide us with some answers.

First, where is the routine LOGLU described in the manuals? We have been unable to discover any references to the routine and would appreciate a description of its usage. We are using RTE-II on our system.

Second, it would be extremely beneficial to have a version of the CLEXT program which would work properly for LU's 2 and 3. Can the current version be easily adapted, and if so, how? Or, would you be able to publish another version which would do a more appropriate job for the system LU's? I am sure that there are many other users who could benefit from such a routine.

Third, could you pass on a suggestion that the CLEXT routine modified to work for all disc LU's be included in a future update to RTE-II/III?

Thank you.

Sincerely,

David W. Palmerston
Administrative Assistant
Helical Products Company, Inc.

Dear David,

The routine LOGLU is described in the RTE-IV manual as follows:

LOGLU — RETURNS LU OF SCHEDULING PROGRAM.

This routine returns the logical unit numbers of the terminal at which the currently executing program was scheduled.

The calling sequence in Assembly Language is:

```
                 EXT LOGLU
                      :
                 JSB LOGLU
                 DEF RTN
                 DEF IDUMY
          RTN
                      :
```

The calling sequence in RTE FORTRAN IV is:

```
                LU=LOGLU(ISES)
```

Upon return:

LU = A-register = LU number of device at which program was scheduled.

B-register = ASCII LU number

ISES = Dummy value reserved for future use by HP.

Comments:

Note that LOGLU must be called as a function. LOGLU will return the LU number of the console from which the currently executing program was scheduled. This LU number is passed down from the Father program to the Son program when one program schedules another program for execution. If the program was scheduled by interrupt or from the time list, the scheduling LU will be LU 1, the system console.

The answer to your second question regarding the program CLEXT is that yes, it can be modified to work properly for LU's 2 and 3. Simply, remove line 42,

```
      IF(DLU.EQ.2.OR.DLU.EQ.3) GO TO 10
```

so that no check will be made for these LU's.

However, since LU's 2 and 3 cannot be packed there is another modification you may want to make to CLEXT so that there will be more usable space on the disc. Keep in mind that when a file is purged the space released can only be used by a file of size smaller than or equal to the purged file. If the new file is smaller than the purged file then chances are the difference in size will be left as unused space. Eventually, of course, the disc cartridge will have to be packed to obtain this leftover space. So, we recommend using a standard file size (24 blocks is a good size) and allowing extents to be created for files above this size. This standardization of file size allows space to be reused easily and reduces the amount of time-consuming cartridge packing required. The block size of the new file created by the program CLEXT is set in BUFFST's DATA statement. On line 30 the 2H-1 should be changed to 2H24 or whatever size you feel is optimal for your system.

Lastly, HP no longer makes enhancements to RTE-II/III. RTE-II is a mature product; it is subject to regular maintenance, but is not enhanced. RTE-III is obsolete software; it is no longer actively marketed by HP.

Thank-you for your interest and support. Keep those letters coming!

Sincerely,

Samantha

DS/1000 TIME GETTER

Bob O'Leary/Airesearch Manufacturing Co.

Normally, after bootstrap the user must enter the date and time in the system. However, in a nodal system the date and time can be extracted from one of the other active nodes in the network. The program that follows is a utility to help users initialize a node in a DS/1000 network. This utility will run from the WELCOM file of a node, after LSTEN has enabled the node's links to the rest of the network. It will poll the other nodes until it finds one with a reasonable date and time; it will then enter that time into the local system via the message processor. This can be very useful in networks where it is important to have the various time-of-day clocks synchronized with each other.

The user should modify the "NODES" array to reflect the nodes in his N.D.T.; there should be active (non-zero) transaction timeouts entered for each node in the network, so that a "dead" node will not hang up the system. If the utility cannot find an active node with a reasonable time-of-day, it will print out a message on the system console to that effect. And lastly, the routine needs, of course, SSGÅ access. Good luck.

```
PAGE 0001  FTN.   9:45 AM  SAT., 18  AUG., 1979

FTN4,L
      PROGRAM TINIT
C
C     PROGRAM TO INITIALIZE TIME-OF-DAY AT BOOTSTRAP TIME
C     THIS MODULE POLLS THE REMOTE NODES (IN 'NOTES')
C     UNTIL HE FINDS A NODE WITH A REASONABLE TIME-OF-DAY.
C     HE THEN ENTERS THIS TIME IN HIS LOCAL NODE
C     THROUGH THE LOCAL MESSAGE PROCESSOR.
C     NOTE---THIS PROGRAM REQUIRES SSGA ACCESS.
C     R.L.O'LEARY---4/19/79
C
      DIMENSION MESG(10),NODES(2),IT(5)
      DIMENSION MSON(6)
C
      DATA MSON/ 2HON,2H,U,2HPL,2HIN,2H,N,2HOW /
C
      DATA MESG/ 2HTI,2HME,2H N,2HOT,2H I,2HNI,2HTI,
          2HAL,2HIZ,2HED /
      DATA MSLEN/ 10 /
      DATA NODES/ 3, 2 /
      DATA NNOD/ 2 /
C
C
      DO 99 I=1,NNOD
      CALL DEXEC(NODES(I),11+100000B,IT,IYR)
      GO TO 99
  98  IF(IYR.GE.1979) GO TO 125
  99  CONTINUE
C
C   NO LUCK---NO TIME AVAILABLE.
C
      CALL EXEC(2,1,MESG,MSLEN)
      GO TO 150
C
C   SUCCESS---WE FOUND A VALID TIME-OF-DAY.
C
```

```
 125  CONTINUE
      CALL TIME(NODES(I))
C   RE-ACTIVATE "UPLIN" (DS/1000 WATCHDOG).
      CALL MESSS(MSON,12)
 150  CALL EXEC(6)
C
      END


 FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00091      COMMON = 00000


PAGE  0002  FTN.   9:45 AM  SAT., 18  AUG., 1979


C
C
C
      SUBROUTINE TIME(NODE1)
C
C       SUBROUTINE TO PICK UP TIME-OF-DAY IN A REMOTE NODE,
C            AND ENTER IT INTO THE LOCAL NOTE.
C       NODE1 = REMOTE NODE WHERE CORRECT TIME IS KEPT.
C          SYSTEM LIBRARIES REQUIRED---->
C                          %DSLB2, %DSLB1, %MSYLB.
C          **NOTE** THIS ROUTINE NEEDS SSGA ACCESS.
C                R.L.O'LEARY----4/5/79.
C
      DIMENSION IBUFF(15)
C
      DATA IBUFF/2HTM,2H, ,2HTI,12*2H  /
      DATA ICOMA/ 26000B /
C
C
      CALL DMESS(NODE1,IBUFF(3),2)
      IBUFF(5) = IOR(IAND(IBUFF(5),377B),ICOMA)
      IBUFF(7) = IOR(IAND(IBUFF(7),377B),ICOMA)
      IBUFF(9) = IOR(IAND(IBUFF(9),377B),ICOMA)
      IBUFF(11) = IOR(IAND(IBUFF(11),377B),ICOMA)
      CALL MESSS(IBUFF,24)
C
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00081      COMMON = 00000
```

IDCHK UTILITY PROGRAM

Don Pottenger/DSD

As system manager it is sometimes necessary to know who is grabbing the temporary ID segments. This especially becomes important when all the available ID's have been taken and the system manager has to decide who must go. IDCHK will help you by giving you a list of all the temporary ID's currently in the system. After listing both the long and short ID's, it also prints the number still available.

As an additional option, by specifying the second parameter non-zero, you can get track availability of the cartridges in your cartridge list. This feature is needed less often, but sometimes it is nice to know which CRN has the most available disc space. To schedule IDCHK type:

```
RU,IDCHK,LU,OPTION
```

where LU	=	output device
OPTION	=	If non-zero, print track availability info also.

```
FTN4,L,B
C
C    THIS PROGRAM PRINTS OUT ALL THE TEMPORARY ID SEGMENTS IN THE
C    SYSTEM.  IT ALSO CATEGORIZES THEM INTO LONG AND SHORT ID'S
C
      PROGRAM IDCHK,3,70), ID SEGMENT CHECK 790625
      DIMENSION IP(5),NAME(8,3)
      DIMENSION ISTAT(125),IBUF(16)
      EQUIVALENCE (IP(1),LU),(IP(2),ITKCK)
C
C   GET OUTPUT DEVICE
C
      CALL RMPAR(IP)
      IF(LU.EQ.0)LU=1
C
C   SET COUNTERS AND FLAGS
C
      ILCNT = 1
      ISCNT = 0
      ISHORT = 0
      IEOF = 0
C
C    GET ADDRESS OF KEYWORD BLOCK
C
      KEYWD=IGET(1657B)
C
C    PRINT HEADING
C
      WRITE(LU,5)
  5   FORMAT (7X"TEMPORARY ID SEGMENTS LISTED BY PROGRAM "
     1"NAME & TYPE"//16X"<<<<<<  LONG ID SEGMENTS  >>>>>>")
C
C  SEARCH ID SEGMENTS AND CHECK FOR TEMPORARY ONES
C
 10   DO 50 I=1,8
 20      IDSEG = IGET(KEYWD)
C
C       CHECK FOR END OF KEYWORD BLOCK
C
```

```
             IF (IDSEG.NE.0) GO TO 21
                IEOF = 1
                GO TO 22
  21         CONTINUE
C
C           RECORD NAME
             NAME(I,1) = IGET(IDSEG+12)
             NAME(I,2) = IGET(IDSEG+13)
             NAME(I,3) = IGET(IDSEG+14)
C
C           LOOK FOR SHORT ID
C
             LONG = IAND(NAME(I,3),20B)
C
C           IF FIRST SHORT ID, PRINT LONGS & SET ISHORT(FLAG) TO 1
C
             IF (LONG.EQ.0.OR.ISHORT.EQ.1) GO TO 25
  22            IF (I.EQ.1) GO TO 23
                WRITE(LU,55) ((NAME(J,K),K=1,3),J=1,I-1)
  23            IF (IEOF.EQ.1) GO TO 60
                WRITE(LU,24)
  24            FORMAT(/16X"<<<<<<  SHORT ID SEGMENTS  >>>>>>")
                ISHORT = 1
C
C              CONTINUE LOOP FOR SHORT ID'S
                GO TO 10
   25        CONTINUE
C
C           LOOK FOR BLANK ID'S
C
               IDBLNK = IAND(NAME(I,1),177400B)
               IF (IDBLNK.NE.0) GO TO 40
C
C                COUNT BLANKS AND CONTINUE
C
                  IF (ISHORT.EQ.0) ILCNT = ILCNT + 1
                  IF (ISHORT.EQ.1) ISCNT = ISCNT + 1
   30             KEYWD = KEYWD + 1
                  GO TO 20
C
C            IF NOT BLANK, CHECK IF TEMP. OR PERM.
C
   40          ITEMP = IAND(NAME(I,3),200B)
               IF (ITEMP.EQ.0) GO TO 30
C
C           MAKE THIRD WORD PRINTABLE
C
             NAME(I,3) = IOR(IAND(NAME(I,3),177400B),40B)
             KEYWD = KEYWD + 1
   50        CONTINUE
C
C           WRITE THE EIGHT ONES WE GOT
C
               WRITE (LU,55) ((NAME(I,J),J=1,3),I=1,8)
   55          FORMAT (" "8(3A2,2X))
               GO TO 10
C
C    PRINT OUT AVAILABILITY OF LONG AND SHORT ID'S
C
```

```
 60    WRITE (LU,70)
 70    FORMAT (/14X"<<<<<<  ID SEGMENTS AVAILABLE  >>>>>>")
       WRITE (LU,80) ILCNT,ISCNT
 80    FORMAT (22X,I2" LONG"6X,I2" SHORT")
C
C     DID YOU ASK FOR TRACK AVAILABILITY INFO?
C
       IF (ITKCK .EQ. 0) GO TO 999
       WRITE (LU,90)
 90    FORMAT(/7X"<<<<<<  USER CARTRIDGE TRACK AVAILABILITY  >>>>>>"/
      &46X,"LAST TRACK")
C
C   GET CARTRIDGE LIST
C
       CALL FSTAT (ISTAT)
       DO 130 I=1,120,4
       ISEC = 0
       CALL ASCII(ISTAT(I+2),IASCII)
C
C   IF COMPILED WITH DEBUG DO RTE-IVA STUFF
C
D      IF (ISTAT(I).EQ.2) ISEC = 14
C
C     READ THE DIRECTORY
C
       REG = EXEC (1,ISTAT(I),IBUF,16,ISTAT(I+1),ISEC)
C
C     AND OUTPUT THE INFO
C
       WRITE (LU,100) ISTAT(I),ISTAT(I+2),IASCII,IBUF(10),ISTAT(I+1)
100    FORMAT (9X"LU "I3" CR"I6" ="A2"  NEXT TRACK"I4"  ("I4")")
120    IF (ISTAT(I+4).EQ.0) GO TO 140
130    CONTINUE
C
C     PAGE EJECT
C
C
140    CALL EXEC(3,LU+1100B,-1)
C
 999   END
C
C     THIS SUBROUTINE DETERMINES IF THERE IS A PRINTABLE
C     ASCII EQUIVALENT FOR THE ID'S SECURITY CODE
C
       SUBROUTINE ASCII(BINARY,IA)
       INTEGER BINARY
       IA = IAND(BINARY,377B)
       LBYTE = IAND(BINARY,77400B)
C
C     IF BINARY IS NOT A VALID ASCII CHARACTER,
C     SET IA TO BLANK
C
       IF (IA.LT.40B.OR.IA.GT.176B) IA = 40B
       IF (LBYTE.LT.20000B) LBYTE = 20000B
       IF (LBYTE.EQ.77400B) LBYTE = 20000B
       IA = IOR(LBYTE,IA)
       RETURN
       END

FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)
**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00051   COMMON = 00000
```

SAMPLE OUTPUT WITHOUT OPTIONAL PARAMETER

```
        TEMPORARY ID SEGMENTS LISTED BY PROGRAM NAME & TYPE

                  <<<<<<  LONG ID SEGMENTS  >>>>>>
FTN4     ASMB     XREF     %BX00    %BX01    %BX02    %BX03    TEDIT
JEDIT    EDITR    DL       CMM4     SAM      LGTAT    PRTSV    IDCHK
FLUSH    SWAPT    HALT     FMG63    FMG57    LOA56    FMG56    EDT57
FMG61    FMG51    FMG01    FMG52    TEST6    QUE58    FMG58    QPRNT
RUN51    JED61    EDT53    FMG53    IDC56    ..CHK    QAR

                  <<<<<<  SHORT ID SEGMENTS  >>>>>>
ACCT1    ACCT2    ACCT3    ACCT4    ACCT5    F4.0     F4.1     F4.2
F4.3     F4.4     F4.5     ASMB0    ASMB1    ASMB2    ASMB3    ASMB4
BASC1    BASC2    BASC3    BASC4    BASC5    BASC6    BASC7    BASC8
SED1     SED2     SED3     QY       QY00     QY01     QY02     QY03
QY04     QY05     QY06     QY07     QY08     QY09     QY10     QY11
QY12     QY13     QY14     QY15     QY16     QY17     QY18     QY19
QY20     QY21     QY22     QY23     QY24     QS       QS00     QS01
QS02     QS03     QS04     QS05     QS06     QS07     QS08     QS09
QS10     QS11     QS12     QS13     QS14     QS15     QS16     RTML1
RTML2    RTML3    RTML4    RTMG1    RTMG2

                <<<<<<  ID SEGMENTS AVAILABLE  >>>>>>
                       41 LONG      23 SHORT
```

```
                 OUTPUT WITH OPTIONAL PARAMETER NON-ZERO


            TEMPORARY ID SEGMENTS LISTED BY PROGRAM NAME & TYPE

                      <<<<<<  LONG ID SEGMENTS  >>>>>>
FTN4     ASMB     XREF     %BX00    %BX01    %BX02    %BX03    TEDIT
JEDIT    EDITR    DL       CMM4     SAM      LGTAT    PRTSV    IDCHK
FLUSH    SWAPT    HALT     FMG63    FMG57    LOA56    FMG56    EDT57
FMG61    FMG51    FMG01    FMG52    TEST6    QUE58    FMG58    QPRNT
RUN51    JED61    EDT53    FMG53    MTD01    IDC56    ..CHK    QAR

                      <<<<<<  SHORT ID SEGMENTS  >>>>>>
ACCT1    ACCT2    ACCT3    ACCT4    ACCT5    F4.0     F4.1     F4.2
F4.3     F4.4     F4.5     ASMB0    ASMB1    ASMB2    ASMB3    ASMB4
BASC1    BASC2    BASC3    BASC4    BASC5    BASC6    BASC7    BASC8
SED1     SED2     SED3     QY       QY00     QY01     QY02     QY03
QY04     QY05     QY06     QY07     QY08     QY09     QY10     QY11
QY12     QY13     QY14     QY15     QY16     QY17     QY18     QY19
QY20     QY21     QY22     QY23     QY24     QS       QS00     QS01
QS02     QS03     QS04     QS05     QS06     QS07     QS08     QS09
QS10     QS11     QS12     QS13     QS14     QS15     QS16     RTML1
RTML2    RTML3    RTML4    RTMG1    RTMG2

                    <<<<<<  ID SEGMENTS AVAILABLE  >>>>>>
                          40 LONG       23 SHORT


             <<<<<<  USER CARTRIDGE TRACK AVAILABILITY  >>>>>>
                                                      LAST TRACK
               LU  25 CR    25 =    NEXT TRACK  85   ( 201)
               LU  34 CR 21319 =SG  NEXT TRACK  20   ( 201)
               LU  11 CR   255 =    NEXT TRACK 486   ( 499)
               LU  28 CR   254 =    NEXT TRACK 164   ( 201)
               LU  33 CR 21323 =SK  NEXT TRACK 192   ( 201)
               LU  35 CR 17228 =CL  NEXT TRACK  33   ( 201)
               LU   2 CR     2 =    NEXT TRACK 254   ( 255)
               LU   3 CR     3 =    NEXT TRACK 216   ( 255)
               LU  13 CR    13 =    NEXT TRACK 253   ( 255)
               LU  18 CR 32767 =    NEXT TRACK 125   ( 201)
               LU  12 CR 23456 =[   NEXT TRACK 224   ( 255)
```

LOADING PROGRAM SEGMENTS FROM FMP FILES

Larry W. Smith/HP Fullerton

Are you a user suffering from an overdose of segmentation? Well, if you are a victim of this potentially crippling convenience, I might have just the remedy for you. This article will review some existing system restrictions and reveal how it is possible to overcome the limitation of the total number and disc location of program type 5 segments. In specific, we will explore a means of loading any number of segments from any disc lu into the segment overlay area of an executing program. I think that you will find the method described in this article both simple and flexible to use.

HISTORICAL BACKGROUND

This capability was developed for a customer that markets an extremely sophisticated three-dimensional graphics manufacturing design software package. In order for them to maintain standard ANS FORTRAN coding for implementation on a variety of mini-computer systems and due to the complexity of their software, segmentation is inevitable. The total number of segments that are required to run a plot is about 320. This presents somewhat of a problem to RTE since it has a maximum of 255 ID segments for all program types, including segments. This was the beginning of SLOAD.

REVIEW OF EXISTING SYSTEM LIMITATIONS

All executable programs and segments in the RTE system must reside on either LU=2 or LU=3 of the system disc area prior to being dispatched into a partition for execution. This is primarily due to the internal table structure of ID segments. An ID segment is used by RTE to keep track of a program's activity (status, priority, etc.) as it requests resources during Real-Time. You might think of ID segments as a very efficient way of cataloging executable programs. In addition, assuming there is enough disc space to hold all segments, the total number of all programs and/or segments is limited to 255. Furthermore, the total number of tracks available to hold all these type programs is 512 minus the system size and any FMP tracks on LU=2 and/or LU=3. All in all, it appears that a desirable compromise would be to limit the number of segments and discover other means, such as scheduling other programs that share system common, as an alternative solution.

To summarize system program and segment limitations, we have the following:

1. Must reside on LU=2 or LU=3 in the work area or SP'd into an FMP file.
2. Maximum limit of 255 total programs and/or segments.
3. Maximum number of tracks to hold all programs and/or segments is 512 minus system size and FMP area.

SOLUTION: LOADING SEGMENTS FROM FMP FILES

To overcome the three above restrictions, a subroutine could be written in FORTRAN (or Assembly language) that would essentially open a file, read all records into the segment overlay area, close the file, and finally jump to the beginning of the segment. The advantages and disadvantages of this approach could be summarized as follows:

Advantages

1. Segment can be located on any disc LU.
2. Build-in FMP file protection to such things as the 'OF' command or system re-boot.
3. Parameters in the 'SLOAD' call in the same positions as the 'EXEC' call.
4. No significant limitation on the total number of segments.

Disadvantages

1. Load time is slightly slower than using an 'EXEC' call.
2. Caution must be taken when attempting to share segments with more than 7 concurrent programs.
3. There is no provision in the 'SLOAD' and 'FREAD' routines to pass parameters on the call but this could easily be done.

Depending upon the application, the convenience of this method could become the only self-defeating obstacle due to the FMP overhead compared to the system EXEC overhead.

Let's assume this method is feasible for your application. The subroutine responsible for loading the segment is called as follows:

```
CALL SLOAD(ierr, name, isc, icr)
```

ierr	FMP file error return.
name	file name containing SP'd segment.
isc	file security code.
icr	cartridge reference number.

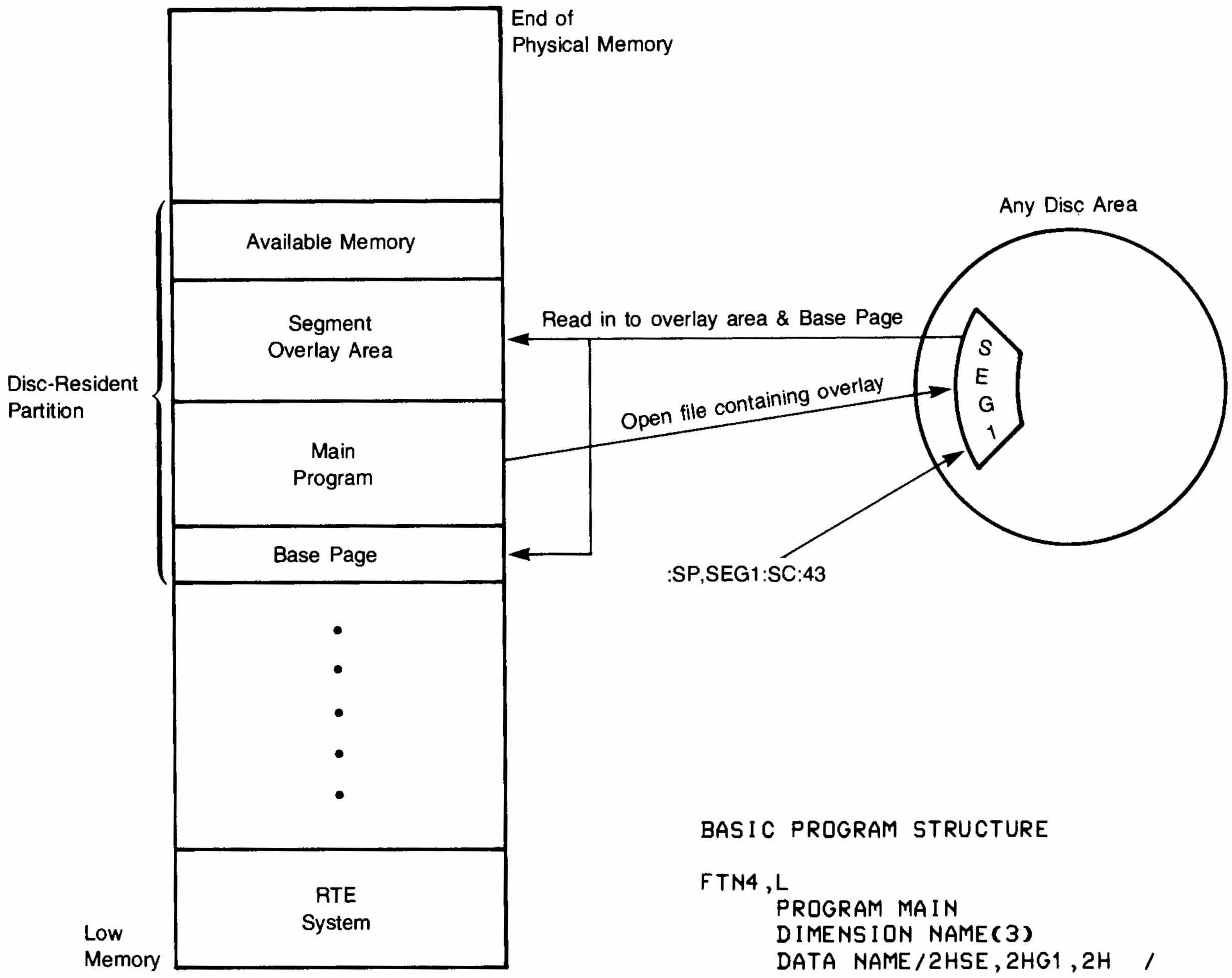
The subroutine would accomplish the following tasks:

1. Open the file.
2. Check for open errors.
3. Read entire main program into the segment overlay area.
4. Check for read errors.
5. Read any base page (if any).
6. Check for read errors.
7. Close the file.
8. Check for close error.
9. Transfer control to start of the segment.

All the rules for segment program coding remain the same with the only difference being in coding a call to 'SLOAD' rather than an 'EXEC' call. In either case, the parameter positions are identical.

To illustrate the mechanics of how this can be done, the following FORTRAN program called 'SLOAD' with one Assembly language subroutine called 'FREAD' and a test case program and segment are listed at the end of this article. The overall picture of this process is as follows:

OVERALL PICTURE



```
BASIC PROGRAM STRUCTURE

FTN4,L
      PROGRAM MAIN
      DIMENSION NAME(3)
      DATA NAME/2HSE,2HG1,2H   /
       .
       .
       .
      CALL SLOAD(IERR,NAME,2HSC,43)
       .
       .
      IF(IERR.LT.0) GO TO 999
       .
       .
       .
      END
      END$
```

Note the compatability
with existing
CALL EXEC(8,NAME)

Figure 1

As can be seen, the only coding difference to the programmer is the usage and interpretation of the parameters in the 'SLOAD' call.

When loading the main program, the SLOAD subroutine must be loaded with the main and not with a segment. As an example, let's assume we had four modules to load as follows:

MAIN — FORTRAN main test program that calls SLOAD.

SEG1 — FORTRAN segment that SLOAD loads for MAIN.

SLOAD — FORTRAN FMP segment loader subroutine.

FREAD — Assembly subroutine called by SLOAD to do actual memory loading.

The loading sequence would be as follows:

```
:RU,LOADR
 /LOADR:   RE,%MAIN::13
  COM    40002 40002
  MAIN   40003 40205  FMP SEGMENT LOADER TEST PROGRAM
 /LOADR:   RE,%SLOAD::13
  SLOAD  40206 40666  FMP SEGMENT LOADER  (LWS,    REV. 1902)
 /LOADR:   RE,%FREAD::13
  FREAD  40667 40742  READ FMP TYPE 6 RECORD INTO MEMORY  (LWS)
 /LOADR:   RE,%SEG1::13

  RMPAR  40743 41001  771116  24998-16001
  OPEN   41002 41167  92002-16006 741205
  CLOSE  41170 41306  92002-16006  771115
  READF  41307 42045  92002-16006 770801
  REIO   42046 42163  92067-16035 REV.1826 780509
  R/W$   42164 42317  92002-16006  740801
  P.PAS  42320 42346  92002-16006 740801
  FMTIO  42347 43765   24998-16002 REV.1805 780303
  FRMTR  43766 46567   24998-16002 REV.1805 780303
  FMT.E  46570 46570   24998-16002 REV.1805 780303
  PNAME  46571 46636   771121  24998-16001
  $OPEN  46637 47045  92002-16006    740801
  RW$UB  47046 47317  92002-16006 750422
  RWND$  47320 47442  92002-16006  771121


SEG1   47443 47467  FMP SEGMENT LOADER TEST SEGMENT
 /LOADR:   END

   5 PAGES RELOCATED  5 PAGES REQ'D    NO PAGES EMA    NO PAGES MSEG

   /LOADR:MAIN   READY

   /LOADR:$END

:SP,SEG1::3
:RU,MAIN,SE,G1,20040B
MAIN RUNNING.
ATTEMPTING LOAD OF FILE 'SEG1  '.
SEG1 IN.
MAIN BACK RUNNING.
:
```

CONCLUSION

This is a method of overcoming the segmentation blues on an RTE system. In RTE-M, a supported program called 'APLDR' will accomplish about the same thing since there is no system disc or program swapping capability. The author has not had time to do performance evaluations or comparisons on this method, but I believe in most applications it will not present a real problem. It might be noted that other means of loading segments such as obtaining actual track and sector location of FMP files or using system level $XSIO calls can be used to improve loading time. This program will eventually be available in LOCUS for a nominal fee.

PROGRAM LISTINGS

```
FTN4,L
C
      SUBROUTINE SLOAD(IERR,FNAME,ISEC,ICR),FMP SEGMENT LOADER
C
C  SUBROUTINE DESCRIPTION
C  ----------------------
C
C    THIS SUBROUTINE LOADS THE CONTENTS OF AN FMP FILE CONTAIN-
C  ING AN SP'D PROGRAM SEGMENT INTO THE SEGMENT OVERLAY AREA OF
C  AN EXECUTING PROGRAM.  THE PROGRAM PERFORMS NECESSARY FMP FILE
C  ERRORS BUT DOES NOT VALIDATE THE CONTENTS OF THE FILE NOR RE-
C  STRICT ITS TYPE.  CAUTION MUST BE USED IN ATTEMPTING TO USE
C  TYPE 0 FILES SINCE THERE IS NO CHECK FOR THIS POSSIBILITY. IT
C  MIGHT ALSO BE NOTED THAT THE APPROPRIATE CONTENTS OF THE FIRST
C  BLOCK OF THE FILE CONTAINING IMPORTANT INFORMATION ABOUT THE
C  SEGMENT'S BOUNDS ARE NOT VALIDATED.  THEREFORE, THE USER MUST
C  ENSURE THAT THE INTEGRITY OF THE TYPE 6 FILE IS BEYOND ANY
C  POSSIBILITY OF CORRUPTION.
C
C  CALLING SEQUENCE
C  ----------------
C
C    CALL SLOAD(IERR,FNAME,ISEC,ICR)
C
C     IERR ----> FMP ERROR RETURN FROM OPEN, READF, OR CLOSE CALL.
C
C     FNAME ---> FILE FNAME CONTAINING THE DESIRED TYPE 5 SEGMENT.
C                THE FILE TYPE CAN BE ANY FILE TYPE BUT IT IS
C                SUGGESTED THAT IT BE A TYPE 6.
C
C     ISEC ----> FILE SECURITY CODE.
C
C     ICR -----> CARTRIDGE REFERENCE NUMBER OR LOGICAL UNIT NUMBER.
C
C
C  EXTERNAL REFERENCES
C  -------------------
C
C     OPEN ----> FMP FILE OPEN ROUTINE.
C
C     READF ---> FMP FILE READ ROUTINE.
C
C     CLOSE ---> FMP FILE CLOSE ROUTINE.
C
C     FREAD ---> ASSEMBLY LANGUAGE SUBROUTINE TO READ ONE RECORD
C                FROM AN FMP FILE INTO MEMORY.
C
```

```
C     SJUMP ---> ASSEMBLY LANGUAGE ROUTINE THAT TRANSFERS CONTROL
                 TO THE SEGMENT OVERLAY AREA.
C
C     BPMOV ---> ASSEMBLY LANGUAGE ROUTINE THAT MOVES A BLOCK OF
C                MEMORY FROM LOCAL BUFFER 'INBUF' ONTO THE
C                PROGRAM'S BASE PAGE AREA.
C
C  VARIABLE NAME USAGES
C  --------------------
C
C  FNAME       FMP FILE NAME CONTAINING TYPE 6 SP'D SEGMENT.
C
C  DCB         FMP DATA CONTROL BLOCK ADDRESS.
C
C  INBUF       INPUT BUFFER ADDR FOR EACH RECORD FROM TYPE 6 FIL.
C
C  DCB1        POINTS TO FIRST WORD OF DATA CONTROL BLOCK.
C
C  INBF1       POINTS TO FIRST WORD OF INPUT BUFFER.
C
C  INBF8       POINTS TO PRIMARY ENTRY PT. IN SGMT OVERLAY AREA.
C
C  INBF23      POINTS TO SEGMENT LOW MAIN ADDRESS.
C
C  INBF24      POINTS TO SEGMENT HIGH MAIN ADDDRESS.
C
C  INBF25      POINTS TO SEGMENT LOW BASE PAGE ADDRESS.
C
C  INBF26      POINTS TO SEGMENT HIGH BASE PAGE ADDRESS.
C
C  IDADR       CONTAINS SEGMENT'S PRIMARY ENTRY PT. FOR EXECUTION
C              WHICH IS GIVEN TO SUBROUTINE 'SJUMP'.
C
C  ADDRES      CONTAINS THE CURRENT MEMORY ADDRESS WHEN LOADING
C              THE MAIN SEGMENT BODY AND BASE PAGE.
C
C  PRGLEN      CONTAINS THE MAIN SEGMENT BODY LENGTH.
C
C  BPLEN       CONTAINS THE BASE PAGE LENGTH.
C
C  RECORD      CONTAINS THE CURRENT RANDOM ACCESS RECORD NO. WHEN
C              LOADING BLOCKS FROM THE FILE INTO THE SEGMENT
C              OVERLAY AREA.
C
C  LENR        CONTAINS THE ACTUAL REQUESTED RECORD LENGTH WHEN
C              READING RECORDS FROM THE FILE INTO MEMORY.
C
C  LEN         CONTAINS THE FMP RETURNED RECORD LENGTH WHICH
C              SHOULD EQUAL THAT OF VARIABLE 'LENR'.
C
C  IER         CONTAINS THE LAST FMP CALL ERROR.
C
C
C  WARNINGS
C  --------
C
C  1. THIS SUBROUTINE DOES NOT ATTEMPT TO PROCESS USER TYPE ERRORS
C     WHICH COULD BE CLASSIFIED AS 'CORRUPTABLE' TYPE ERRORS.
C
C  2. THIS SUBROUTINE MUST BE LOADED WITH THE MAIN PROGRAM ONLY.
C
```

```
C
      IMPLICIT INTEGER (A-Z)
C
      DIMENSION FNAME(1),DCB(16),INBUF(128)
      EQUIVALENCE(DCB1,DCB(1))
      EQUIVALENCE(INBF1,INBUF(1)),(INBF8,INBUF(8))
      EQUIVALENCE(INBF23,INBUF(23)),(INBF24,INBUF(24))
      EQUIVALENCE(INBF25,INBUF(25)),(INBF26,INBUF(26))
C
C... ATTEMPT FILE OPEN & CHECK FOR FILE ACCESSIBILITY.
C    FORCE OPEN TO NON-EXCLUSIVE, UPDATE, & TYPE 1 ACCESS ...
C
      CALL OPEN(DCB1,IERR,FNAME,7,ISEC,ICR)
      IF(IERR)999,30
C
C... FILE IS READY - READ FIRST RECORD TO GET PROG. BOUNDS ...
C
   30 CALL READF(DCB1,IER,INBF1,128,LEN,1)
      IF(IER)666,50
   50 IDADR=INBF8
      ADDRES=INBF23
      PRGLEN=INBF24-ADDRES+1
      BPLEN=INBF26-INBF25+1
C
C... LOAD MAIN PROGRAM (RECORDS 2-(N-1)) INTO OVERLAY AREA ...
C
      RECORD=2
      LENR=128
   82 IF(PRGLEN.LT.128) LENR=PRGLEN
      CALL FREAD(DCB1,IER,ADDRES,LENR,LEN,RECORD)
      IF(IER)666,85
C
C... UPDATE MEMORY ADDRESS & NEXT BLOCK ...
C
   85 PRGLEN=PRGLEN-128
      IF(PRGLEN.LE.0) GO TO 45
      ADDRES=ADDRES+128
      RECORD=RECORD+1
      GO TO 82
C
C... LOAD BASE PAGE AREA ...
C
   45 ADDRES=INBF25
      RECORD=RECORD+1
      LENR=128
   90 IF(BPLEN.LT.128) LENR=BPLEN
      CALL READF(DCB1,IER,INBF1,BPLEN,LEN,RECORD)
      IF(IER)666,93
   93 CALL BPMOV(INBF1,ADDRES,LEN)
      BPLEN=BPLEN-128
      IF(BPLEN.LE.0) GO TO 94
      ADDRES=ADDRES+128
      RECORD=RECORD+1
      GO TO 90
C
C... CLOSE FILE & TRANSFER CONTROL TO SEGMENT ...
C
   94 CALL CLOSE(DCB1,IER)
      IF(IER)666,95
C
```

```
C... DIVE INTO SGMT WITH A `JSB 0,I' & HOPE FOR GOOD WEATHER ...
C
   95 CALL SJUMP(IDADR)
C
C... RETURN POINT IN CASE A `JMP ADRES,I' IS DONE IN SEGMENT ...
C
      GO TO 999
C
C... RETURN TO CALLER IF FILE OPEN, READF, OR CLOSE ERRORS ...
C
  666 IERR=IER
  888 CALL CLOSE(DCB1)
C
  999 END

  FTN4 COMPILER: HP92060-16092 REV. 1805 (780310)

  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00305      COMMON = 00000
```

```
                    ASMB,R,L,Q  ** READ FMP TYPE 6 RECORD INTO MEMORY **
**  NO ERRORS PASS1 **RTE ASMB 92067-16011**

                    ASMB,R,L,Q  ** READ FMP TYPE 6 RECORD INTO MEMORY **
*
00000                     NAM FREAD,7 READ FMP TYPE 6 RECORD INTO MEMORY
*
                          ENT FREAD,SJUMP,BPMOV
*
                          EXT .ENTR,READF,$LIBR,$LIBX,.MVW
*
*  SUBROUTINE DESCRIPTION
*  ----------------------
*
*    THIS SUBROUTINE IS RESPONSIBLE FOR ACCOMPLISHING THREE TASKS
*  AS REQUESTED BY THE FMP SEGMENT LOADER `SLOAD':
*
*    FREAD --->  READS A RECORD FROM 1 TO 128 WORDS IN LENGTH FROM
*                AN FMP FILE INTO THE SPECIFIED MEMORY ADDRESS AS
*                PASSED BY THE CALLER IN `IADRS'.
*
*    SJUMP --->  TRANSFERS CONTROL TO THE SEGMENT OVERLAY AREA BY
*                EXECUTING A `JSB 0,I' INSTRUCTION USING THE MEMORY
*                ADDRESS AS PASSED BY THE CALLER IN `IDADR'.
*
*    BPMOV --->  MOVES A BLOCK OF DATA AS PASSED BY THE CALLER IN
*                `IBUFR' ONTO BASE PAGE ADDRESS SPECIFIED IN `IDRES'.
*
*
```

```
*  CALLING SEQUENCES
*  ----------------
*
*    CALL FREAD(IDCB,IER,IADRS,IL,LEN,NREC)
*
*      IDCB ----> DATA CONTROL BLOCK ADDRESS.
*
*      IER -----> FMP ERROR RETURN
*
*      IADRS ---> MEMORY ADDRESS (SAME AS BUFFER ADDRESS).
*
*      IL ------> REQUEST LENGTH.
*
*      LEN -----> RETURNED RECORD LENGTH.
*
*      NREC ----> RECORD NUMBER.
*
*
*    CALL BPMOV(IBUFR,IDRES,NWORD)
*
*      IBUFR ----> BUFFER ADDRESS CONTAINING THE DATA TO BE MOVED.
*
*      IDRES ----> STARTING MEMORY ADDR WHERE DATA IS TO BE MOVED.
*
*      NWORD ----> NUMBER OF WORDS TO BE MOVED.
*
*
*   CALL SJUMP(IDADR)
*
*      IDADR ----> MEMORY ADDRESS TO WHICH CONTROL IS TO BE GIVEN.
*
*
*  WARNINGS
*  --------
*
*  1. THIS SUBROUTINE DOES NOT ATTEMPT TO PROCESS USER TYPE ERRORS
*     WHICH COULD BE CLASSIFIED AS 'CORRUPTABLE' TYPE ERRORS.
*
*  2. THIS SUBROUTINE MUST BE LOADED WITH THE MAIN PROGRAM ONLY.
*
*
 00000 000000  IDCB  NOP
 00001 000000  IER   NOP
 00002 000000  IADRS NOP
 00003 000000  IL    NOP
 00004 000000  LEN   NOP
 00005 000000  NREC  NOP
*
 00006 000000  FREAD NOP  <<< ENTRY >>>
 00007 000001X       JSB .ENTR     RETRIEVE CALLERS PARAMETERS.
 00010 000000R       DEF IDCB
*
```

```
0079  00011 000002R       LDA IADRS,I   GET STARTING MEMORY ADDRESS
0080  00012 000017R       STA INBUF    AND SET-IN AS ACTUAL BUFFER ADDR
0081  00013 000002X       JSB READF     INITIATE
0082  00014 000023R       DEF *+7        FMP RECORD
0083  00015 100000R       DEF IDCB,I      READ
0084  00016 100001R       DEF IER,I        DIRECTLY
0085  00017 000017R INBUF DEF *             INTO
0086  00020 100003R       DEF IL,I           SPECIFIED
0087  00021 100004R       DEF LEN,I           MEMORY
0088  00022 100005R       DEF NREC,I           ADDRESS.
0089  00023 000006R       JMP FREAD,I   RETURN TO CALLER.
0090*
0091  00024 000000  IDADR NOP
0092  00025 000000  SJUMP NOP  <<< ENTRY >>>
0093  00026 000001X       JSB .ENTR
0094  00027 000024R       DEF IDADR
0095  00030 000024R       LDA IDADR,I    GET SEGMENT JUMP ADDRESS.
0096  00031 000000        JSB 0,I        DIVE INTO SEGMENT & AND PRAY.
0097*
0098  00032 000025R       JMP SJUMP,I   IN CASE THE SEGMENT DOES THIS.
0099*
0100  00033 000000  IBUFR NOP           SOURCE BUFFER ADDRESS.
0101  00034 000000  IDRES NOP           DESTINATION MEMORY ADDRESS.
0102  00035 000000  NWORD NOP           NUMBER OF WORDS TO MOVE.
0103  00036 000000  BPMOV NOP  << ENTRY >>>
0104  00037 000001X       JSB .ENTR
0105  00040 000033R       DEF IBUFR
0106  00041 000003X       JSB $LIBR     TURN OFF INTERRUPT SYSTEM.
0107  00042 000000        NOP
0108  00043 000033R       LDA IBUFR     GET SOURCE ADDRESS.
0109  00044 000034R       LDB IDRES,I   GET DESTINATION ADDRESS.
0110  00045 000005X       JSB .MVW      GO MOVE THE BLOCK.
0111  00046 100035R       DEF NWORD,I
0112  00047 000000        NOP           FOR FIRMWARE COMPATABILITY.
0113  00050 000004X       JSB $LIBX     RETURN
0114  00051 000052R       DEF *+1        TO
0115  00052 000053R       DEF *+1         RTE.
0116  00053 000036R       JMP BPMOV,I   RETURN TO CALLER.
0117*
0118                      END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

```
FTN4,L
      PROGRAM MAIN(3),FMP SEGMENT LOADER TEST PROGRAM
      COMMON IRET
      DIMENSION NAME(5)
      CALL RMPAR(NAME)
      WRITE(1,100)
  100 FORMAT("MAIN RUNNING.")
      ASSIGN 10 TO IRET
      WRITE(1,105)NAME(1),NAME(2),NAME(3)
  105 FORMAT("ATTEMPTING LOAD OF FILE `"3A2"'.")
      CALL SLOAD(IERR,NAME,2HSC,-3)
      IF(IERR)77,10
   77 WRITE(1,101)IERR
  101 FORMAT("FMP FILE SEGMENT OPEN ERROR="I6)
   10 WRITE(1,103)
  103 FORMAT("MAIN BACK RUNNING.")
      END

 FTN4 COMPILER: HP92060-16092 REV. 1805 (780310)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00131      COMMON = 00001
```

```
FTN4,L
      PROGRAM SEG1(5),FMP SEGMENT LOADER TEST SEGMENT
      COMMON IRET
      WRITE(1,100)
  100 FORMAT("SEG1 IN.")
      GO TO IRET
   99 END

 FTN4 COMPILER: HP92060-16092 REV. 1805 (780310)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00021      COMMON = 00001
```

CONVERSION OF COMPUTER PROGRAMS TO HP1000 RTE-IV

Jack B. McAlister/Technology Development Corporation

INTRODUCTION

The conversion of computer programs will enable the HP 1000 RTE-IV user to adhere to the adage of "Not re-inventing the wheel". There are computer programs that have already been developed on other computer systems to perform the same operations as many of those desired under the RTE-IV operating system.

The conversion of computer programs from other computer systems to a form that is acceptable on the HP 1000 RTE-IV operating system requires several phases of analysis and conversion. In the following discussion of these various phases, source program refers to the originating computer system program and converted program refers to the RTE-IV program.

SOURCE LANGUAGE

The first item of program conversion is to assess the language of the source program. Programs written in languages other than FORTRAN or BASIC are difficult, at best, to convert. Generally, BASIC programs require very little conversion. Therefore, attention is focused on conversion of FORTRAN programs.

A careful analysis of the source program is required to ascertain the type of FORTRAN used. The HP 1000 RTE-IV FORTRAN IV is based on the 1966 ANSI standard. FORTRAN 77 permits several different types of FORTRAN statements, including "IF ELSE THEN", and special DO forms, e.g., "REPEAT". Some FORTRAN compilers also have non-ANSI standard statements or extensions that don't exist in the HP 1000 RTE-IV FORTRAN IV.

SEGMENTATION

The conversion process should, if possible, begin with a compiled listing of the source program. On most computer systems, the FORTRAN IV compiler listing will include the core size requirements for the source program. These requirements are an aid in the determination of segmentation requirements on RTE-IV.

To segment a program, break the source program into logical or functional modules. Each of these modules can be converted into a segment. Large subroutines are prime candidates to become segments. Also, look for blocks of code within large subroutines that with little or no change can be converted into subroutines.

Since the return to a segment is always at the top of the segment, determine whether any segments will be re-entered after completion of a call to another segment. In such cases, a variable(s) in COMMON can be used to control the jump to the statement following the call.

EXAMPLE

```
       COMMON ---------,NSWTCH,---------
        ....
        ....
        ....
       IF (NSWTCH.EQ.0) GOTO 50
       GOTO (100,200,300),NSWTCH
    50 CONTINUE
        ....
        ....
        ....
       NSWTCH=1
       CALL EXEC (8,SEG1)
   100 CONTINUE
        ....
        ....
        ....
       NSWTCH=2
       CALL EXEC (8,SEG2)
   200 CONTINUE
        ....
        ....
        ....
       NSWTCH=3
       CALL EXEC (8,SEG3)
   300 CONTINUE
        ....
        ....
        ....
        ....
```

COMMON BLOCK

If the converted program is to be segmented and the source program has multiple labeled COMMON blocks, these blocks must be merged into one COMMON block. All segments must have the same COMMON allocation for sharing of data between segments. The parameters list of any subroutine that is being converted into a segment should be included in the COMMON block. It is also useful to include spare variables and arrays for future use. If this is not done, the programmer will have to be very careful when one of the segments requires a change in the COMMON block. An easy method of ensuring the correct COMMON allocation is to have the COMMON block as a separate FMP disc file which can be merged into the segment sources when a change is required.

SUBROUTINES

A large number of statements in a subroutine or segment can make analysis, understanding, conversion, and debugging difficult. This type of subroutine can be broken into two or more subroutines as follows:

a. Determine logical sections of the subroutine.

b. Use the EDITOR to break and form new subroutines using the determinations from step a. Do not attempt to resolve or declare variables or arrays at this time. Add the FTN4 control statement only.

c. Compile and cross reference the new sections.

d. Use the cross reference list to prepare a parameters list for communications between the new section and the calling subroutine.

WORD SIZE

Another item requiring careful scrutiny is the word size of the source program computer system. Computer system word sizes vary from 16-bits to 32-bits, 36-bits, or even 60-bits. The following types of statements must be analyzed and adjusted for these differences:

1. FORMAT
2. ARRAY DECLARATIONS
3. DATA STATEMENTS
4. EXECUTABLE STATEMENTS USING ARRAYS
5. DO LOOP RANGES

READS & WRITES

All formatted READ statements must be checked for the use of ERR=nnn and/or END=nnn. Device status check statements must be included after the READ to satisfy ERROR and END branches.

Also, careful attention must be given to the type of READ or WRITE. For example, is a spool file or a disc file being used? If either of these conditions are true, then the procedures described in the Batch Spool Monitor manual are to be used.

If spool and/or disc files are to be used, an analysis of the file contents is required to determine possible file length. The size of the files should be specified as large as practical since these files cannot have more than 255 extents.

DEBUGGING

Aids for debugging large segmented programs on the HP 1000 RTE-IV are very important. The use of the HP 'DEBUG' package requires an assembly language equivalent listing which would be very large and time-consuming; and an understanding of the HP assembly language is required. This requirement can be alleviated through the use of a form of tracing. The use of a "TRACE" variable, in COMMON, has been found to be one of the best debugging aids. This variable is set in the RUN parameters and retrieved with the "RMPAR" routine. The programmer inserts "IF" statements in the segments to write information that will aid in the flow analysis. When the program has been debugged and is operational, the TRACE statements can be left in the program as an aid for future maintenance. The TRACE variable will aid the programmer in the duplication and verification of the users anomoly during the maintenance phase.

The use of multiple levels of tracing is also valuable. By assigning different values of the TRACE variable to separate segments, modules, subroutines, etc., specific detailed information may be traced. The assignment of one value for the trace variable will enable a scan of total program flow.

Example

```
NTRACE = 99 FOR TRACE ALL MODULES
       = 88 FOR TRACE ROOT MAIN
       =  1 FOR TRACE SEGMENT 1
       =  2 FOR TRACE SEGMENT 2
       =  3 FOR TRACE SEGMENT 3
          ....
          ....
          ....
       =  0 FOR NO TRACE
```

To enable checkout to begin with the completion of source code for the first segment, all other segments should be included as "STUBS" (Dummy segments that return immediately, furnishing plausible values for their output parameters). In this way the entire program appears to be operational from the beginning. As subsequent segments are coded, they can replace their Dummy "Representatives" and testing can continue. This process is in accordance with TOP-DOWN STRUCTURED Programming.

TIMING

Timing is another very important consideration during the conversion of a large program. During code conversion, the programmer should be observant of sections of code that could be time "bottlenecks". When the debugging is in process these sections of code can be observed closely and a "true time user" can then be made more time efficient with a different coding scheme. Code that reads "well" may not be time efficient. Disc accesses and sorts are usually areas that deserve close scrutiny.

CONCLUSION

Program conversion is a way of benefiting from development work done by others. Not only do you profit from others design knowledge, but you also save yourself the time you would have spent coding and debugging. Quantitatively, if the original development required 1/2 man-year to place into operation and the program can be converted in 2 man-months, there is a savings of 4 man-months.

The processes discussed in this article have been used by the author to convert a program which requires a 1008K-byte partition on a large computer. The converted program requires an RTE-IV 48K-byte partition and consists of a root main and 18 segments. The original programming was accomplished in 18 man-months and the conversion in 6 man-months. For a program that provides a valuable product the time spent on the conversion is usually recovered quickly.

THE FUNDAMENTALS OF HP-IB ADDRESSING

Neal Kuhn/DSD,Applications Development

HP-IB is a powerful interfacing concept that allows multiple instruments to be connected to the same bus. This concept helps to economize the system cost with respect to cabling, interface hardware, and user software effort. Programming devices with HP-IB is straightforward; however, if full utility is desired a person implementing an HP-IB system should understand how HP-IB addresses are created and used. The purpose of this article is to describe the HP-IB addressing scheme, and discuss how it is used in an HP 1000 Computer System.

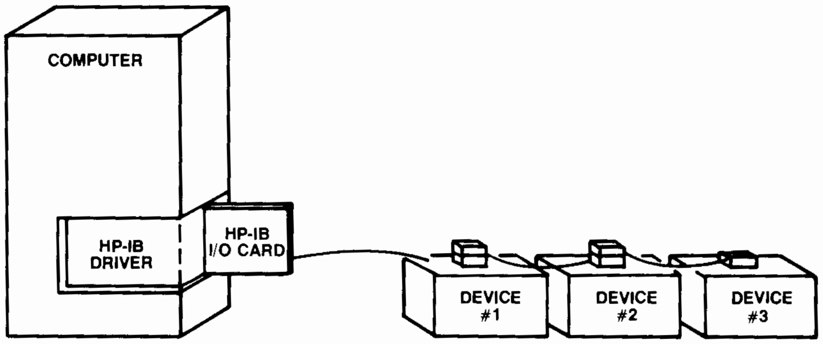


Figure 2

Figure 1 illustrates the typical connection pattern for a group of HP-IB instruments. The bus itself is a multidrop arrangement in which each device has its own address. The address is set with switches which are usually located on the back of the instrument, or with jumpers which are usually on the circuit board associated with the instrument's I/O. Each manufacturer installs the switches or jumpers in a different place, and labeling is not always clear. Therefore, care should be taken to ascertain which switch is the most significant bit, and which position is "1" or "0". Application Note Series 401 contains information regarding the setting of the HP-IB address switches for many of the HP-IB instruments and peripherals. A typical HP-IB address grouping is shown in Figure 2.

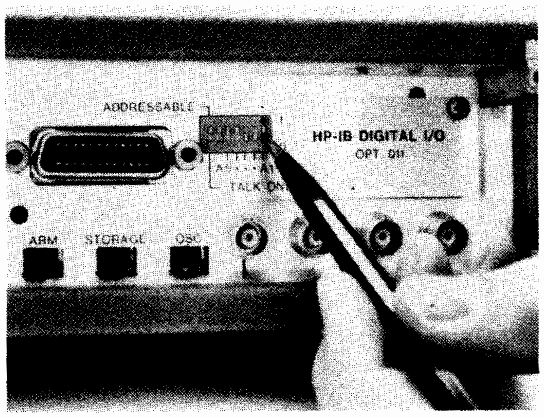


Figure 1

Five address bits are used to obtain the binary pattern for up to 32 device addresses, but only 30 different settings can be used for instruments. One address is withheld by the controller for use as its own address, since the I/O card itself is an HP-IB device. For the HP 1000, this address is address 0. HP-IB address 31 (37B) is defined by the HP-IB for the special purpose of telling everyone to stop talking (untalk) or stop listening (unlisten). All devices respond to the untalk and unlisten commands regardless of their addresses. Addresses 1 to 30 (36B) may be arbitrarily assigned to your device.

When the HP-IB's ATN (attention) management line is asserted, information is sent out from the controller over the bus data lines. This information tells the devices which HP-IB command to perform, or which device should talk and which devices should listen during a transfer of data. A coding scheme is established for HP-IB where a parity bit and two data bits are appended to the five bit address to form an eight bit "command" byte. The parity bit is not usually used. A device looks for its address in the command byte, and also looks at the two extra bits to determine what the command was. The choices for these two bits are:

WORD	MEANING
00 XXXXX	This is a bus "universal" command and xxxxx is the command to perform. The universal commands are used to clear devices, trigger them, and to set up polling sequences for service requests.
01 XXXXX	This is a "listen" command, and device xxxxx is to listen to the bus for data.
10 XXXXX	This is a "talk" command, and device xxxxx is to send data onto the bus.
11 XXXXX	This is "secondary" command, and xxxxx is a device dependent function code for the currently addressed device.

When the command bits are combined with the device address or function code, a seven bit ASCII character is formed. Note for universal commands that the characters are non-printing. Table 1 illustrates the ASCII characters formed for talker and listener combinations of device address and command bits.

For an example, if the ASCII command string "_?5L" is sent onto the bus from the controller, everyone would untalk (_), everyone would unlisten (?), the device set to address 5 is told to listen for data (5), and the device set to address 12 is told to talk (L). After the ATN management line is de-asserted, device 12 will begin talking, device 5 will listen, and everyone else will ignore the transaction. Note that device addresses 5 and 12 were arbitrarily selected.

According to the IEEE-488 standard, a device that is made a listener will continue to listen until it is made a talker, or the unlisten command is sent.* Assigning a new listener will not unlisten the previous one. This is the technique used by HP-IB to assign multiple listeners. The active talker will change when it is made a listener, or another device is made a talker, or the untalk command is sent. To assure that the proper devices are talking and listening, the convention usually adopted is to always send the untalk and unlisten commands first, and then set up the appropriate talkers and listeners.

Communication with devices over HP-IB can be accomplished in two different ways. One method is referred to as "automatic addressing", and the other is referred to as "direct addressing". Automatic addressing is a simple and straightforward technique which can be used for a majority (if not all) of HP-IB transactions.

Automatic addressing can utilize the standard READ,WRITE/PRINT contexts and formatting procedures. For READ statements, it is assumed that the HP 1000 is the listener, and the addressed device is the talker. For WRITE and PRINT statements, the HP 1000 is the talker and the addressed device is the listener.

There are other techniques to unlisten or untalk an instrument (such as interface clear), but they are unaesthetic, and are used as bail out measures. Also, the untalk for my listen address is optional, but normally used. The only time that this option is not used is with terminals where the keyboard (Talker) and the display (Listener) are both active.

Table 1. ASCII Character Equivalents for Talker and Listener Values

COMMAND BITS	SWITCH SETTINGS	OCTAL VALUE OF SWITCHES	LISTENER (XX=01)	TALKER (XX=10)
XX	0 0 0 0 0	0	SPACE	@
XX	0 0 0 0 1	1	!	A
XX	0 0 0 1 0	2	"	B
XX	0 0 0 1 1	3	#	C
XX	0 0 1 0 0	4	$	D
XX	0 0 1 0 1	5	%	E
XX	0 0 1 1 0	6	&	F
XX	0 0 1 1 1	7	'	G
XX	0 1 0 0 0	10	(	H
XX	0 1 0 0 1	11	)	I
XX	0 1 0 1 0	12	*	J
XX	0 1 0 1 1	13	+	K
XX	0 1 1 0 0	14	,	L
XX	0 1 1 0 1	15	-	M
XX	0 1 1 1 0	16	.	N
XX	0 1 1 1 1	17	/	O
XX	1 0 0 0 0	20	0	P
XX	1 0 0 0 1	21	1	Q
XX	1 0 0 1 0	22	2	R
XX	1 0 0 1 1	23	3	S
XX	1 0 1 0 0	24	4	T
XX	1 0 1 0 1	25	5	U
XX	1 0 1 1 0	26	6	V
XX	1 0 1 1 1	27	7	W
XX	1 1 0 0 0	30	8	X
XX	1 1 0 0 1	31	9	Y
XX	1 1 0 1 0	32	:	Z
XX	1 1 0 1 1	33	;	[
XX	1 1 1 0 0	34	<	\
XX	1 1 1 0 1	35	=	]
XX	1 1 1 1 0	36	>	^
XX	1 1 1 1 1	37	?	_

NOTE: The two shaded values have special uses. Address "0" is used by the computer. Address 37B is used by HP-IB to UNTALK or UNLISTEN all devices.

The Device Reference Table (DRT) structure in the HP 1000 lends itself well to HP-IB programming. When a READ, WRITE, or PRINT statement is executed, the LU specified is converted into the respective EQT number of the HP-IB and a subchannel number. The subchannel number is the HP-IB device's address (1 to 36B). The HP-IB driver "automatically" converts the subchannel number into the proper HP-IB talker or listener code and performs the following:

For a READ statement:

1. ATN is asserted by the HP 1000
2. UNTALK (_) is sent
3. UNLISTEN (?) is sent
4. The TALKER character for the specified LU is sent (see table 1)
5. ATN is de-asserted
6. The device talks and the HP 1000 listens

For a WRITE or PRINT statement:

1. ATN is asserted by the HP 1000
2. UNTALK (_) is sent
3. UNLISTEN (?) is sent
4. The LISTEN character for the specified LU is sent
5. ATN is de-asserted
6. The HP 1000 talks and device listens

The following statement illustrates an example of how automatic addressing is used. If a device requires the characters "T3" to initiate a measurement, the statement:

```
      WRITE (28,101)
  101 FORMAT ("T3")
```

could be used if the device was assigned to LU 28. Unformatted (free field format) and binary readings can also be used.

There are times when it is desirable to continue a READ or WRITE function without altering talk or listen addresses. This can be accomplished by using an LU number that points to the bus itself (subchannel 0). Since talking or listening to subchannel 0 would mean talking to itself, the system assumes that you wish to perform the I/O transaction without disturbing prior addressing. The I/O task is performed without any command information (asserting ATN). This technique is useful for some buffered devices when a portion of the buffer is sent each time without re-addressing. Also, this technique can speed processing when the talkers and listeners do not need to be changed.

The direct addressing technique is a faster and more efficient method than auto-addressing, but places the burden of bus control on the user program. Under direct addressing, the user program provides two buffers to the HP-IB driver. One buffer contains the command information which will be sent over the bus while ATN is asserted. This is used to assign the appropriate talkers and listeners, or to send universal and secondary command characters. The other buffer either contains the data to be sent out, or is the place where incoming data will be stored. Remember to specify the computer's talk or listen address with direct addressing if the computer will participate in the data transfer.

There are two ways to directly address devices over the bus. HP-IB library routines CMDR and CMDW (command read and write) or EXEC calls can be used in FORTRAN. In BASIC, CMDR and CMDW must be used, since EXEC calls cannot be used.

The HP-IB Users Guide (HP part number 59310-90064) explains how to use CMDR and CMDW, but two vital points are often overlooked. First, CMDR and CMDW were intended for use with BASIC, and expect the command and data buffers to contain character strings. In FORTRAN, the format of these buffers must resemble a string. This means that the first word of the buffer must contain the number of characters in the string. After this first word, each additional word will contain two characters. The second point to remember is that direct addressing requires that the LU of the bus be used. The HP-IB driver checks to see if the subchannel specified is 0. If it isn't, an error occurs, the task is rejected, and the program is terminated.

The library commands CMDW and CMDR reformat the buffers and make calls to EXEC. If the language used allows calls to EXEC, they can be used directly. The format for the direct addressing EXEC call is:

```
        CALL EXEC(ICODE,ICNWD,IDBFR,IDLEN,ICBFR,ICLEN)
```

where:

ICODE = Function Code

1 = READ
2 = WRITE

ICNWD = Control word containing the bus LU and an indicator for type of transfer

10000B+LU = ASCII data record with an end of record indicator.

10100B+LU = Fixed length binary record with an end of record indicator.

12000B+LU = ASCII data record without an end of record indicator.

12100B+LU = Fixed length binary record without an end of record indicator.

IDBFR = Name of data buffer

IDLEN = Length of data buffer in either characters (bytes) or words.

n = words
−m = characters
0 = no data

ICBFR = Name of buffer containing the HP-IB commands

ICLEN = Length of command buffer

n = words
−m = characters
0 = no command buffer used

Note in the above EXEC call that the command and data buffers are optional parameters. If the EXEC call is to perform HP-IB commands only, no data buffer is used. The EXEC call expects these parameters to be specified, so remember to supply arguments for both the buffer name and length (set the values to 0's). If the talk and listen assignments are not altered, the command buffer is not used. It is either set to 0's, or can be omitted. If it is omitted, be sure to turn off bit twelve in the above control words (subtract 10000B from each of the commands).

Special bus related commands can also be sent to the HP-IB bus to clear devices, to trigger them, and to regulate polling for service requests. These commands, which are referred to as "universal" commands, are shown in table 2. They can be sent in command buffers under direct addressing, or with library routines. The HP-IB Users Guide explains the library calls in detail, and gives examples of their use.

The preceeding discussion covered addressing and talking to HP-IB devices. Extra processing power and efficiency can be obtained by utilizing a full understanding of the addressing scheme. Remember, it is important to know your address when getting on the bus.

Table 2. HP-IB Universal Commands

COMMAND	ACRONYM	OCTAL CODE	FUNCTION
Local Lockout	LLO	21	Causes all responding devices to disable their front panel local-reset buttons. Devices need not be addressed.
Device Clear	DCL	24	Causes all responding devices to return to a pre-determined state. Devices need not be addressed.
Selected Device Clear	SDC	04	Causes the current addressed devices to re-set to a predetermined state.
Group Execute Trigger	GET	10	Causes all current addressed devices to initiate a preprogrammed action.
Go To Local	GTL	01	Causes the currently addressed devices to return to local control.
Serial Poll Enable	SPE	30	Establishes serial poll mode, such that all co-operating devices, when addressed, will pro-vide status information. An ensuing read will take a single 8-bit byte of status.
Parallel Poll Configuration	PPC	05	Assigns an HP-IB DIO line to a cooperating device for the purpose of responding to a parallel poll.
Parallel Poll Unconfiguration	PPU	25	Resets all parallel poll devices to a prede-termined condition.
Take Control	TCT	11	This command is used to pass active control-ler function to another device on the bus. This function may NOT be used under the constructs of the current driver.
The remaining unspecified codes are reserved for future use and should not be used indiscriminately in any control buffer. This will avoid future difficulties.			

REMOTE DATA BASE ACCESS: HOW DOES IT WORK?

Carol Jonas/DSD, Technical Marketing

If you have been reading the literature released recently regarding the new IMAGE/1000 product, you will notice that Remote Data Base Access is being called the most significant enhancement to IMAGE/1000. A statement of this type is usually meant to precipitate questions such as "What exactly is Remote Data Base Access?" and "How do I use it?". Perhaps a more important question, from a programmer's point of view is "How does it work?". It is the purpose of this article to answer this last question.

Remote Data Base Access (RDBA) is a mechanism within IMAGE/1000 which allows for accessing a data base on a remote machine in a Distributed Systems/1000 network as easily as accessing a data base on your local machine. The DS/1000 software/firmware and special IMAGE/1000 software are combined to provide this feature. In order to explain the special IMAGE software in this article, it is necessary to assume the reader has a basic understanding of DS/1000 and local IMAGE/1000.

With local IMAGE/1000, manipulating an existing data base can be performed in several different ways; using programmatic calls, QUERY, or DBBLD. Manipulation of a data base through RDBA can be performed in two different ways; either with programmatic calls or with QUERY. The RDBA programmatic calls provide the same functions as the local calls with nearly identical calling sequences. The only exception is opening a data base with the DBOPN call. With RDBA the number of the node at which the data base resides must be specified along with the data base name, security code, and cartridge reference number. In addition, the RDBA calls can be used to manipulate a local data base by specifying the local node number, or a negative one, in the DBOPN call. This provides for greater program flexibility and transportability. A program can be written which can be run at any node in a DS/1000 network and can access a data base on any node in the network.

Using QUERY to access a data base on a remote node is performed by scheduling QUERY through a REMAT RU or RW command at the node at which the data base resides. QUERY does not perform RDBA calls but rather, when run interactively, performs remote I/O. When scheduling QUERY on the remote node, you pass it the number of your node. QUERY then uses that node number when performing I/O to direct its prompts and responses to, and accept input from, the terminal which you are using. You can also schedule QUERY on a remote node to run in batch mode. In batch mode, QUERY will expect the input file to reside on the node at which it is executing. QUERY does not perform remote file access calls. However, these two modes can be mixed. For instance, you can schedule remote QUERY with a batch input file and a logical unit number for an output device. In this case, QUERY would expect the input file to be at its node, but would direct its output to the LU at the node specified in its run string. The general rule is QUERY will perform remote I/O to a device but not to a file. Remote QUERY can be considered as a user program which uses DEXEC calls to do I/O.

As you can see, the implementation of RDBA through QUERY was done in a fairly straightforward fashion. There are several reasons for this implementation, including its simplicity, over any others which require RDBA and Remote File Access (RFA) calls. RDBA calls were not used in QUERY because they could generate an inordinate amount of line traffic. For instance, if a FIND command which required a serial read of a data set with 30,000 entries was entered in QUERY, using RDBA calls, 30,000 DBGET requests would have to be sent down the line and 30,000 replies to those requests received. An additional consideration in not using the RDBA calls, as well as the main reason for not using RFA calls, in QUERY was the size of the actual subroutines. QUERY already takes approximately 32K bytes in code space alone and is broken into 26 segments. For remote I/O an additional 2K bytes of code space is used plus SSGA must be mapped into the partition. For either RDBA or RFA calls, another 2K to 4K bytes of code space would be added. At that rate, QUERY would quickly become unusable in the average RTE-IVB system.

While the implementation for remote QUERY is relatively straightforward, programmatic RDBA is more involved. Programmatic RDBA was implemented along a scheme that is very similar to DS/1000's Program-to-Program (PTOP) communication. The scheme includes subroutines which initiate the RDBA requests and can be considered the master routines, and two monitors, or slave programs, which service the requests and send replies. The RDBA subroutines are the user interface to RDBA that was mentioned earlier. A program which makes RDBA calls can be considered akin to a PTOP master program.

The RDBA monitors are RDBAM and RDBAP. RDBAM (Remote Data Base Access Monitor) is essentially a traffic controller like PTOPM. When an RDBA request is received by the DS/1000 software, it is sent to RDBAM via class I/O in the way that any DS/1000 request is channeled to the proper monitor. However, unlike most DS/1000 monitors, RDBAM does not service the request directly but rather passes it to a copy of the second RDBA monitor for servicing.

The second RDBA monitor, RDBAP (Remote Data Base Access Program), is the program which does the actual data base manipulation and initiates replies to the requests. RDBAP can be considered akin to a PTOP slave program. There is one copy of RDBAP for each program which uses the RDBA master routines. It is part of RDBAM's function to create a copy of RDBAP for each master program when it first opens a remote data base and to release that copy when the master program closes its last remote data base. RDBAM coordinates the creation and deletion of RDBAP copies as well as the routing of RDBA requests with a table kept in SSGA called RD.TB. There is room in this table for 20 entries where each entry corresponds to one master program-RDBAP copy pair. The entry for each pair contains information to uniquely identify the pair and a class number for the RDBAP copy along which RDBA requests are passed to the copy.

The following figure illustrates the flow of information and control for an RDBA request, excluding the initial data base open which causes the creation of the RDBAP copy and the final data base close which causes the deletion on that copy.

Using figure 1 as a guide, the general processing of an RDBA call would be as follows. When an RDBA call is executed, control passes to the RDBA subroutine (1). The subroutine picks up the parameters passed to it and does some initial validity checking on those parameters. For instance, the value of the mode parameter is checked to make sure it is within the proper range for this call. The subroutine then uses the information in the parameters to build a DS/1000 request buffer (2). This buffer contains everything the software on the remote machine needs to rebuild the data base request as well as routing information for the RDBA and DS/1000 software. This buffer is then passed to the DS/1000 software (3). The DS/1000 software/firmware sends the request to the remote machine (4) and the RDBA program is put into general wait state on a class number awaiting a reply from the remote machine.

On the remote side, the DS/1000 software picks up the request buffer (5), sees that it is an RDBA request, and sends it to RDBAM via class I/O (6). RDBAM retrieves the request from its class and then, based on the information in the buffer and RD.TB, determines which of a possible number of copies of RDBAP will service this request (7). RDBAM reroutes the request

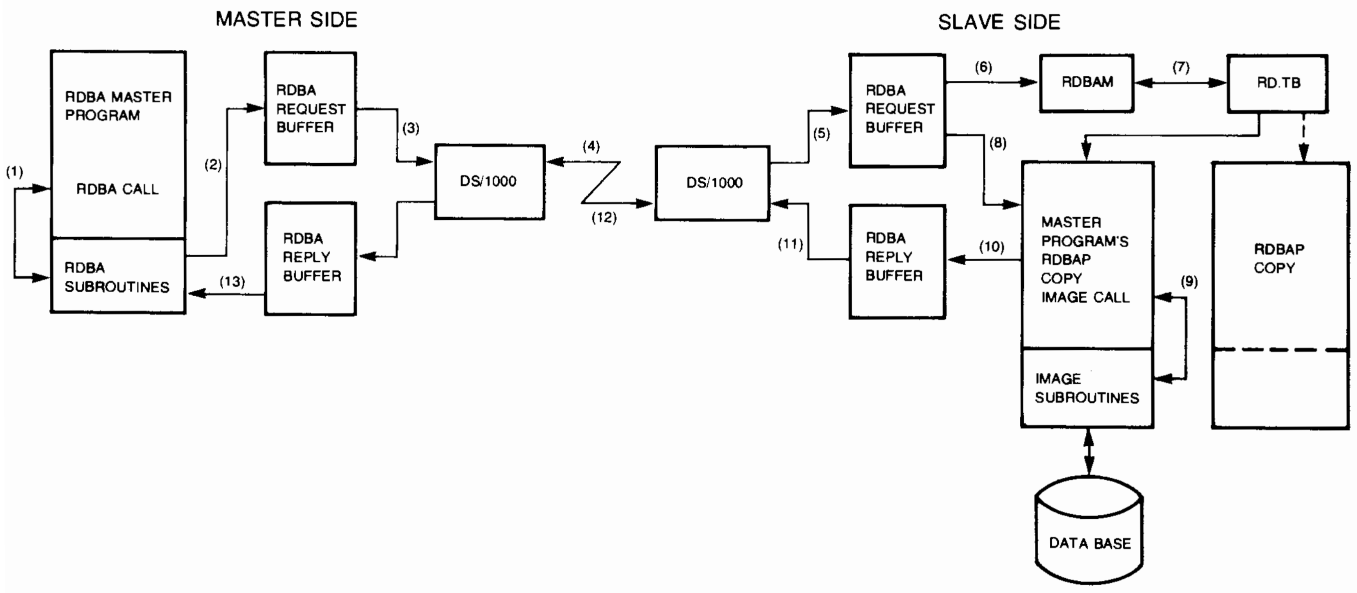


Figure 1

buffer via class I/O to the proper RDBAP copy (8) and then puts itself into general wait state on its class with a class GET awaiting another RDBA request. The RDBAP copy retrieves the request buffer from its class, uses the information contained in the buffer to rebuild the original data base request, and then performs the IMAGE call locally (9). When control returns from the local IMAGE subroutine, RDBAP builds a DS/1000 reply buffer (10) to send back to the requesting program. This reply buffer contains all information returned from the IMAGE subroutine, such as the status array, and DS/1000 routing information. The reply buffer is passed to the DS/1000 software (11) and RDBAP puts itself into general wait state on its class awaiting another RDBA request to service. The DS/1000 software sends the reply buffer back to the requesting program (12) where the RDBA software retrieves it (13). The buffer is then separated into the return parameters from the original RDBA call and control is passed back to the calling program (1).

As with all general descriptions, there is more going on than is stated. To better understand the workings of programmatic RDBA, it might be worthwhile to know about some of its data structures. During local IMAGE processing, for each data base opened by a program, an area in its partition is used to hold a Run Table which contains complete information about the data base, such as all the items' names, types, and lengths, all the sets' names, types, the items in each set, and path information for each set. This is also true for RDBA. What is normally called the Run Table is built in the RDBAP copy's partition since it is just another program accessing the data base. There is a shortened version of this Run Table built in the master program's partition. This shortened version consists only of a data base control block, item table, set table, and a sort table. The data base control block contains global information about the data base. The item table consists of an entry for each item accessible by the program where each entry contains the item's name, number, and length. The set table consists of an entry for each set accessible by the program where each entry contains the set's name, number, entry length, and, if the set is a master, the number of its key item. The sort table is a table of pointers into the item table and set table, sorted alphabetically on the item and set names. It is used for fast access to the item and set table entries.

During local IMAGE operation, the Run Table is built by the DBOPN subroutine from the root file of the data base. For RDBA, the master program's shortened Run Table is built by DBOPN from the parameters in the calling sequence and data passed back in the reply from the RDBAP copy. After a successful open of a data base, the RDBAP copy uses DBINF calls to build the item and set tables for the master program's Run Table. These item and set tables are passed back as data with the reply to the master program. DBOPN completes the master program's Run Table by using the item and set tables to build the sort table.

There are various uses for the shortened Run Table on the master side. One which was mentioned earlier in the general RDBA call process description is validity checking. Each time an RDBA call is made which references a data item or data set, that reference is checked against the item or set table to verify that the program's level of access to the data base allows access to that item or set. The item table is also used whenever an RDBA call is made which references a data item name list, such as DBUPD. Since that item list is a necessary part of the IMAGE call, it must be sent to the RDBAP copy for its use. However, an item name list can be fairly long since each item in the list requires seven bytes for storing the name and a separation character. The RDBA call uses the item table to transform the item name list into its equivalent item number list which only requires two bytes per item. In this way, the amount of information sent across the line is minimized.

As mentioned earlier, the data base control block contains global information about the data base. One word of information which it contains is the number of the node on which the data base resides. Since the only RDBA call which contains the DS/1000 node number for the data base is the DBOPN call, this node number is saved in the Run Table for use in routing all subsequent RDBA calls. Another piece on routing information kept in the data base control block is the Remote Data Base Number. This number is a combination of the RDBAP copy identifier and the data base number. The RDBAP copy identifier is the ordinal number of its entry in RD.TB. The data base number is an index into a table which contains the addresses of the Run Tables for all data bases open to a particular program. This table is kept by the IMAGE software and is updated whenever a data base is opened or closed.

The Remote Data Base Number is returned to the master program in the reply buffer to a DBOPN request. It is passed in the request buffer for each subsequent RDBA call for the data base. RDBAM uses the half of the number which is the RDBAP copy identifier to quickly determine the copy's entry in RD.TB. The other half is used by the IMAGE subroutines appended to the RDBAP copy to determine to which data base this request applies, out of a possible 20 data bases it has open.

Any copy of RDBAP may have a number of data bases open since each copy corresponds to one master program and any IMAGE program may open more than one data base. If, for example, the master program opened three data bases on the same remote node, then its RDBAP copy on that node would open all three data bases for it. In fact, this was the main reason for the design of a two monitor scheme. Since each open data base requires an amount of space in the user program's partition, a single monitor servicing all master RDBA programs would either be severely limited by memory space or would have to use some other form of memory management such as EMA or disc virtual memory. Both of these alternatives, however, are unacceptable because they either require too many system resources or are too slow.

While this implementation was deemed to be the best of the possible, it still has its problems. What happens, for instance, when a master program aborts without closing its last open data base, leaving its RDBAP copy hung on a class number waiting for it to send a request? The data base(s) opened to the RDBAP copy must be closed and the system resources allocated to it released. The utility RECOV provides this service.

RECOV is run at the node on which the data base resides and will display information about all open data bases. In the case of a data base open to an RDBAP copy, it will use RD.TB to obtain and display the name and node number of the master program associated with the RDBAP copy. You "clean-up" after an RDBA master program by specifying a clean-up of the RDBAP copy corresponding to that master. RECOV has an additional feature of performing remote I/O through DEXEC calls for RDBA. It can be run in the same manner as remote QUERY, displaying information at your terminal and allowing you to clean-up after a program remotely.

As you can see, the implementation of RDBA through IMAGE/1000 is not really very complex. This makes the concept easy to understand and to explain. Hopefully this article has given you a good basic understanding of how RDBA works and some appreciation of the decisions that were made during the implementation process.

INTRODUCING THE NEW IMAGE/1000 DBMS

Mike Scott/DSD

A substantially enhanced verson of IMAGE/1000 Data Base Management System (DBMS) has been introduced. IMAGE/1000 is a complete software package for consolidating individual data files into a single, interrelated data base that can be shared by many different people for a wide variety of purposes. The new IMAGE/1000 product was designed for use in HP 1000 computer systems managed by the new RTE-IVB Operating System. A few examples of the new capabilities are given in this excerpt from the press release regarding the new IMAGE/1000:

REMOTE DATA BASE ACCESS — "Perhaps the single most important enhancement is that Remote Data Base Access is now easy for networks of HP 1000 Computer Systems that use Distributed Systems/1000 (DS/1000) software and firmware. This further demonstrates our commitment to distributed processing and HP's Distributed Systems Network philosophy," according to Roger Ueltzen, Marketing Manager of Hewlett-Packard's Data System Division . . . In a user-written application program, data base access to a remote DS/1000 node is easily accomplished by simply specifying the node number in the DBOPN (data base open) call. QUERY, a facility included with IMAGE/1000 that enables the non-programmer to easily retrieve, alter, and report information using English-like commands, can be executed at a remote DS/1000 node to access data stored in an IMAGE/1000 data base.

LARGER DATA BASE CAPACITY — "Another key enhancement was to substantially increase data base capacity," said Ueltzen. The data base size is now limited only by the total available storage, presently a maximum of 960 Mbytes. A data base can contain up to 50 data sets (files). A data set can be as large as the disc volume upon which it resides — presently a maximum of 120 Mbytes. Data entries (records) within each data set may be up to 4,096 bytes long. Also significant is the fact that there may be up to 16 search keys for each data set, thus allowing for fast data access.

Along with the Remote Database Access and the database size increases, a variety of other enhancements have also been made to IMAGE/1000:

- Maximum ASCII character string length increased from 126 to 255 characters.
- The same data item name can be used in different data sets.
- QUERY access is available in batch, as well as interactive mode.
- Real numbers are fully reported in QUERY reports.
- QUERY reports can contain up to 10 lines of output per data entry instead of 1.
- Multiple databases can now be opened to an application program.
- All database modifications are immediately posted to the disc (rather than temporarily held in a memory buffer) for greater database integrity.

ORDERING INFORMATION

The new 92069A IMAGE/1000 is a different product from the old 92063A IMAGE/1000, and it sells for $3000 (U.S. list price). Moreover, the 92069A IMAGE/1000 is Type 1 software, and thus, there is a 92069R product which gives the customer the right to make one copy of the 92069A software for an additional HP 1000 computer system. The 92069R product sells for $1200, and it does require the previous purchase of the 92069A IMAGE/1000 at full list price (less discounts).

UPGRADE INFORMATION

Customers wishing to upgrade from the old 92063A IMAGE/1000 to the new IMAGE/1000 can do so by purchasing the 92069A product with Option 001 at a net price of $1500. The 92069A product with Option 001 does require a previous purchase of the old 92063A IMAGE/1000, and it does not give a customer the right to purchase the 92069R product.

The old IMAGE/1000 (92063A) and the new IMAGE/1000 (92069A/R) are incompatible. The two main reasons for this incompatibility are: 1) double-word integer addressing is used to allow for more than 32K data entries/data set, and 2) the ten IMAGE/1000 subroutines used for host language access have been modified to have the same calling sequence as the IMAGE/3000 subroutines.

Even with the upgrade discount, the decision as to whether an upgrade should be made must be carefully thought out. A reprogramming effort is required to upgrade old IMAGE/1000 application programs — more than just changing the order of the subroutine parameters is required for many of the subroutines. Therefore, customers should only upgrade to the new IMAGE/1000 under the following conditions:

1. Capabilities that exist only in the new IMAGE/1000 are required either today or sometime in the future.

2. It is understood that reprogramming of existing application programs must be done if these programs are to access the new 92069/A data base.

The old IMAGE/1000 (92063A) becomes a mature software product on July 1, 1979. It will be on the Corporate Price List for 12 months (until July, 1980), after which obsolescence and the five year support life will take place. Software bugs will be fixed during the mature period although there will be no enhancements. The old IMAGE/1000 (92063A) will be supported on RTE-IVB so customers are not forced to simultaneously upgrade the operating system and IMAGE/1000. An old 92063A IMAGE/1000 data base and a new 92069A IMAGE/1000 database can both reside on a RTE-IVB based system (see Reference Manual Appendix for details).

Additional points to remember:

1. Software Support Services 92063A IMAGE/1000 (92063S/T) will not be automatically changed to 92069 services (92069S/T) if an upgrade is ordered. A change order must be placed to cancel 92063S/T services and order 92069S/T services if this is what is wanted.

2. A customer who has previously purchased 92063A IMAGE/1000 and has made several copies for additional systems has two ways of upgrading all his systems: a) Purchase the 92069A at full price (less discounts) plus a 92069R for every additional system that is to be upgraded, or b) Purchase 92069A with Option 001 for each of the systems to be upgraded. The second alternative is actually cheaper up until ten or more systems need to be upgraded. Don't forget that the new IMAGE/1000 (92069A) cannot be copied free of charge like the old IMAGE/1000 (92063A)!

ANNOUNCING DATACAP/1000 SOFTWARE PACKAGE FOR REAL-TIME DATA CAPTURE

Mike Scott/DSD

A completely operational real-time data capture system tailored to the user's application can be installed without programming by using Hewlett-Packard's data capture software package, DATACAP/1000. DATACAP/1000 operates on HP 1000 Computer Systems, managed by the new Real-Time Executive Operating System, RTE-IVB. DATACAP/1000 provides the facilities to design and execute transaction specifications which capture and validate data at its source using HP's new 3075A, 3076A, and 3077A data capture terminals. The ability to retrieve, update and validate data in a data base is provided if the user chooses to also use Hewlett-Packard's IMAGE/1000, Data Base Management software.

DATACAP/1000 provides the capability to easily create specifications for data entry transactions through a simple interactive process on a display terminal. Transaction specifications include entry sequence, method (i.e., keyboard or card/badge readers), validation and storage. Data storage can be selected to be on a disc file, an IMAGE/1000 data base, magnetic tape, or a combination of these techniques. Transaction logging can also be selected to provide an audit trail. DATACAP/1000 also provides documentation for each transaction specification to further aid in providing ease of support and maintenance. A flexible terminal management capability is also provided to control the data capture terminals distributed throughout your plant. Ease of modification and flexibility to adjust to changing needs is an important feature of DATACAP/1000 and an HP 1000 Data Capture System.

A real-time data capture system can eliminate the intermediate steps, frustrating delays and errors experienced using traditional data entry methods. Access to a continuing flow of accurate, timely information can be easily provided using DATACAP/1000 and an HP 1000 system. Information, available as events occur can be used effectively to help improve your operations. Applications such as production quality monitoring, inventory control, and work-in-process control can easily benefit from improved data quality and timeliness. Information from the system can be easily obtained through the use of QUERY, IMAGE/1000's English-like inquiry language, or through users' programs which can also take advantage of the compatible Graphics software and hardware available on the HP 1000. Data captured and processed on one system can easily be shared with other systems by using HP's state-of-the-art Distributed Systems Network capability.

The 3075A, 3076A, and 3077A data capture terminals are flexible and easy to use. The 3076A is wall mountable and the 3075A is the benchtop version. User guidance is provided by the use of prompting lights on a panel easily customized to the user's terminology. User definable function keys can be used to simplify special data entry requirements or transaction selection, or if needed, to provide arithmetic operations on numeric values in a transaction. Data is entered through a numeric or optionally alphanumeric key-pad, an optional Type V badge reader, or an optional multi-function reader which can be configured to read alphanumeric information from punched and mark sense cards or industry Type III badges. Data can be displayed using either a numeric display or an optional alphanumeric display. A printer can also be selected to provide alphanumeric output for user receipts or simple reports with key information from the system. The 3077A is a wall mountable time reporting terminal with a Type V badge reader or optional Multifunction card reader. All of these data capture terminal features are supported by DATACAP/1000. Up to (56) data capture terminals with various configurations can be linked to the HP 1000 and managed by DATACAP/1000 by using one or more simple and low cost communication links. Each link can be up to 5 miles long and uses a single, shielded, twisted pair cable. Modem support can also be provided for remote terminals.

PRICING/DELIVERY — DATACAP/1000 is available for $3,000.00. Each additional copy is available at $1,200.00. First deliveries are expected in September, 1979.

The 3075A, 3076A, and 3077A Data Capture Terminals are priced between $2,090 and $4,230 depending on the selected options.

Since all domestic training information is contained in a separate publication, we will no longer duplicate those schedules in the Communicator. The Computer Systems North American Customer Training Schedules (5953-0841) is published quarterly — June, September, December and March. This booklet is automatically sent to all Communicator subscribers on the Software Subscription Service and all HP training centers worldwide. It is our intention to continue to increase the usefulness of the CSG Schedule by including more information about prerequisites and the classes themselves in an attempt to make it a stand-alone document. International schedules, in-so-far as we receive them, will continue to appear in this publication.

HP SOFTWARE SUPPORT — FLEXIBLE & USER ORIENTED

John Koskinen/DSD

With the introduction of RTE-IVB/Session Monitor I'd like to review our software update and upgrade policies. We have tried to provide policy which can satisfy two classes of users:

1. Users who prefer to stay current and up-to-date with HP software.

2. Users who have dedicated, turnkey, or fairly stable applications that don't necessarily need all the latest software updates or enhancements.

Users in the first class stay current by purchasing either the Customer Support Service (CSS) or Software Subscription Service (SSS). Users in the second class acquire current software via an upgrade option on the base product.

LOW PRICE UPDATES

With RTE-IVB we have a product update policy that allows current and up-to-date users a very easy means to get major enhancements to software. We realize that not every user needs to update his software on the fixed, quarterly, update cycles. This is especially true for a major enhancement. It takes time to evaluate the enhanced software and plan for a change over. In this light the enhanced software is treated as a new product offering with a very significant discount for users on CSS or SSS. The option is available for a limited time only, six months, ending December 31, 1979. Figure 1 shows the update path from RTE-IV(A) to RTE-IVB/Session Monitor.

STANDARD PRODUCT UPGRADES

There are many users who prefer to receive software update notices and purchase enhanced software only on an as needed basis; our update/upgrade policy accommodates that preference as well.

Since our quarterly software updates are generally incremental and not cumulative, the software on an out-of-date system cannot be assumed to have a known compatible configuration. There may, in fact even be some firmware changes to be accounted for. It is important in this case to bring the entire software system up to the most recent version level. This can be done by purchasing the base product along with an upgrade option. The upgrade path for RTE-IVB is also given in Figure 1.

RTE-IVB/SESSION MONITOR UPGRADE/UPDATE PATHS

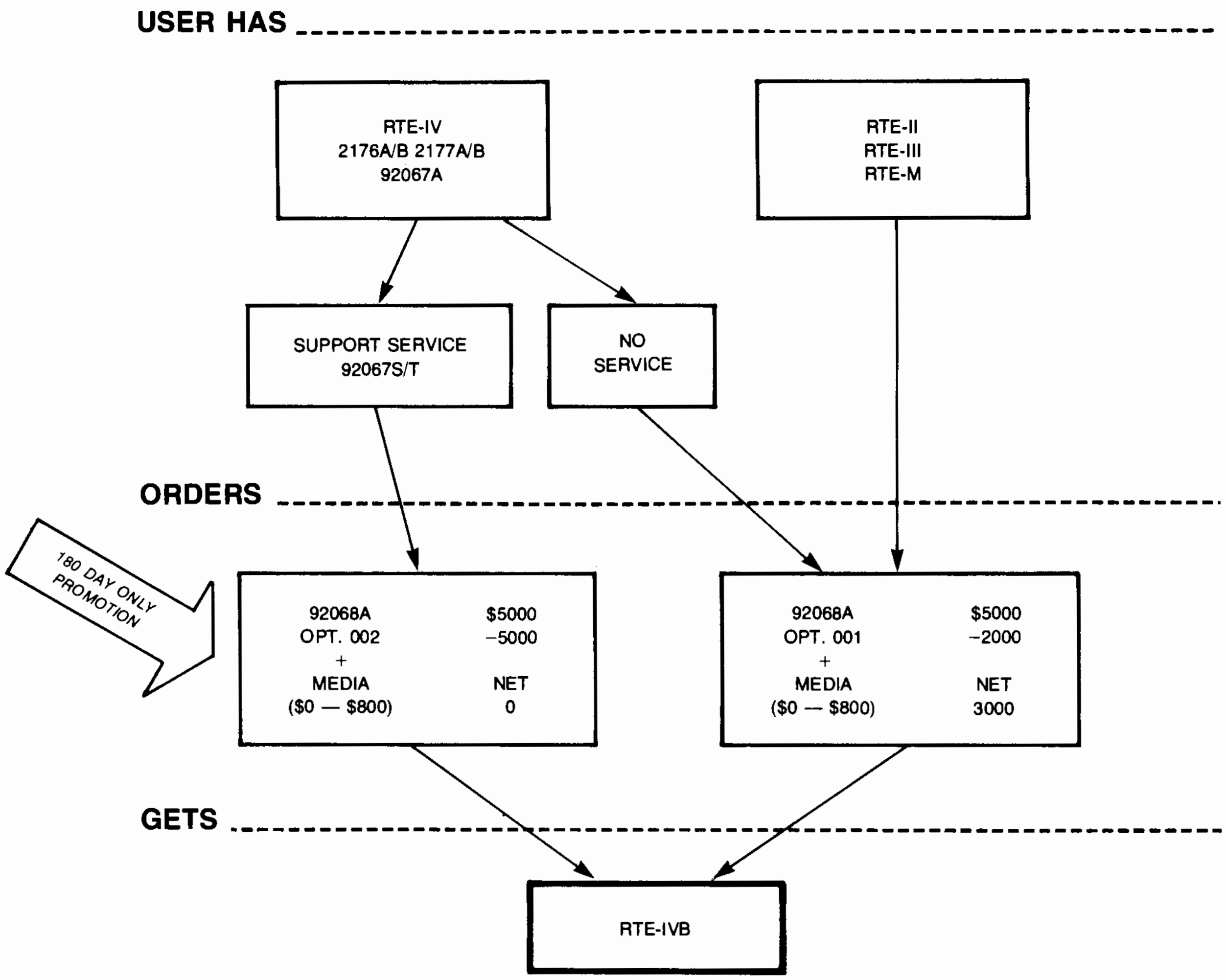


Figure 1

HP 1000 USERS ESTABLISH A USER GROUP

Joe Getkin

On August 23 and 24, the "HP 1000 International Users Group" was born. On these dates, a steering committee of 15 HP users met at Hewlett-Packard headquarters in Cupertino, California. The steering committee personnel represented a cross section of industries using HP 1000's within Canada, Europe, and the United States. The committee began developing a set of by-laws and elected an interim executive board and subcommittee chairmen.

The purpose of the user group is to provide a forum for sharing information among users to enhance their HP 1000 systems, to increase their professional development, and to help reduce their development effort for software, firmware, and systems. The group will also provide a formal communication channel between the membership and Hewlett-Packard.

The users group membership entitles the member to an annual newsletter subscription, an optional user library subscription, voting privileges, and reduced price admittance to the annual convention. This convention will provide both technical and tutorial presentations as well as vendor exhibits.

Hewlett-Packard has indicated that the LOCUS library is being redesigned. When the redesign is complete, which is expected to be late this year, Hewlett-Packard will transfer the library to the users group. The users group will be responsible for maintenance and semi-annual distribution of the complete library to the membership.

The first annual HP 1000 international users group convention is planned to be held in mid 1980. The users group by-laws will be discussed and voted upon. Also, an executive board will be elected to succeed the interim board. Until the meeting is held, the following users will be acting as an interim board of directors and committee leaders.

Interim Board of Directors

President	Stu Troop	General Electric Bridgeport, Conn.
Vice-President	Bert Todtenkopf	Factory Mutual Engineering Norwood, Mass.
Secretary	Paul Miller	Corporate Computer Systems Aberdeen, N.J.
Treasurer	Marvin McInnis	McInnis & Associates Oklahoma City, OK
At Large	Barry Perlman	RCA Labs Princeton, N.J.
At Large	Ron Townsen	Naval Ocean System Center San Diego, Calif.
At Large	Ed Holtzman	Atmospheric Environment Service Downsview, Ontario, Canada

Interim Committee Leaders

Start-Up	Joe Getkin	Ford Motor Co. Dearborn, Michigan
Business Plan	Paul Miller	Corp. Computer Systems Aberdeen, N.J.
Library	Chris Goodey	Becton-Dickinson Salt Lake City, Utah
Publications	Dick Martin	Naval Ocean System Center San Diego, Calif.
Convention	Glen Mortensen	Intermountain Technologies Inc. Idaho Falls, Id.
European	Albert van Putten	Institute for Public Health The Netherlands
By-Laws	Bert Todtenkopf	Factory Mutual Engineering Norwood, Mass.
Nominating	Ron Townsend	Naval Ocean System Center San Diego, Calif.

Like to Help Out?

If you would like to participate in the user group, just contact one of the Board members or committee leaders listed. There's lots of work to be done, and many volunteers are needed!

More Information is Coming

Watch your mail and the next issue of the Communicator/1000 for additional information concerning the users group. The group expects to distribute a brochure/application form, and the first newsletter, within the next two months.

JOIN AN HP 1000 USER GROUP!

Ever wonder how other HP users have used the HP 1000 in application areas similar to your own? Have a special program or driver you'd like to share with other users? Interested in hearing about new developments in HP 1000 hardware and software?

If your answer to any one of these questions is YES, then an HP 1000 user group might be just the thing for you. These and other similar activities are carried on regularly as part of the function of the many HP 1000 user groups that exist around the world. There's a good chance that there's a user group right near your location! To get in on the action, just check the list below, and contact the group nearest you to find out when and where the next meeting will be held.

Or, if there isn't a group near you, why not start one? The Communicator/1000 can help you out by announcing the creation of new groups. Just send a letter — c/o Editor, HP 1000 Communicator, — with the name of your new group and the means by which other users can join. We'll add your group to our list and publish it in the next issue of the Communicator.

Here are the groups that we know of as of August, 1979. (If your group is missing, send the Communicator/1000 editor all of the appropriate information, and we'll update our list.)

NORTH AMERICAN HP 1000 USER GROUPS

Area	User Group Contact	Next Meeting Date	Next Meeting Location
Boston	LEXUS P.O. Box 1000 Norwood, Mass. 02062	Not selected yet (see article on next page about RTE users workshop at URI on October 11, 12)	
Chicago	Dave Olson Institute of Gas Technology 1846 W. Eddy Street Chicago, Illinois 60657	New group — No meeting date selected (312) 248-4017 (Home) (312) 542-7036 (Work)	
New Mexico/El Paso	Gary Galloway Dynalectron Corp. P.O. Drawer O Holleman Air Force Base New Mexico 88330	Sept. 17	—
New York/New Jersey	Paul Miller Corp. Computer Systems 675 Line Road Aberdeen, N.J. 07746 (201) 583-4422	Not selected yet	
Philadelphia	Dr. Barry Perlman RCA Laboratories P.O. Box 432 Princeton, N.J. 08540	Sept 17.	Bell Laboratories Allentown, PA

NORTH AMERICAN HP 1000 USER GROUPS (CONTINUED)

Area	User Group Contact	Next Meeting Date	Next Meeting Location
Pittsburgh	Eric Belmont Alliance Research Ctr. 1562 Beeson St. Alliance, Ohio 44601 (216) 821-9110 X417	Sept. 5 Nov. 7	Holiday Inn Pittsburgh —
San Diego	Jim Metts Hewlett-Packard Co. P.O. Box 23333 San Diego, CA 92123	Not selected yet	
Washington/Baltimore	Paul Toltavull Hewlett-Packard Co. 2 Choke Cherry Rd. Rockville, MD. 20850	Sept. 27	Hewlett-Packard Co. 7121 Standard Dr. Hanover, Maryland
General Electric Co. (GE employees only)	Stu Troupe Special Purpose Computer Ctr. General Electric Co. 1285 Boston Ave. Bridgeport, Conn. 06602	Oct. 1-3	Hewlett-Packard Co. 5201 Tollview Ave. Rolling Meadows, IL.

EUROPEAN HP 1000 USER GROUPS

Area	User Group Contact	Next Meeting Date	Next Meeting Location
London	Rob Porter Hewlett-Packard Ltd. King Street Lane Winnersh, Wokingham Berkshire, RG11 5AR England (734) 784 774	Not selected yet	
Amsterdam	Mr. Van Puten Institute of Public Health Bilthoven Anthony Van Leeuwenhoeklaan 9 The Netherlands	Not selected yet	

HP 1000 SOFTWARE COURSES

Hewlett-Packard offers 15 customer courses to help you learn to use HP 1000 system software. There are six basic courses forming a core from which you can branch out into courses on special products or applications. Figure 1 shows the proper sequences in which to take the courses.

Note that the HP 2645 Terminal Course can be taken as an alternate introduction to the HP 1000 systems. Likewise, the HP 1000 Memory-Based RTE System Course replaces the RTE-IVB Session Monitor Course for those users who have an RTE-M operating system. Descriptions of the available courses are as follows:

22951B Introduction to HP 1000 Computers. A four-day entry level course designed for a beginning programmer who has no hands on experience with a minicomputer, this course will introduce the student to the internal representation of data and machine language instruction as well as architecture and I/O structure. Labs will provide time to write, edit, compile, load and run a simple FORTRAN program using the HP 1000 RTE operating system.

22952B HP 1000 Assembler Programming. This five-day class provides extensive hands-on experience in coding, editing, assembly and debugging of RTE assembler programs using the HP 1000 system. Upon completion, the student will interpret code written using the Assembly Language instruction set and coding format. He will also write, assemble, load and execute main programs and subroutines, interface with high level languages, call subroutines from relocatable library, and use the formatter, exec calls or I/O instructions to write I/O routines. This course requires successful completion of Session Monitor (22994A) OR Memory-Based RTE (22992A).

22961B DS/1000 Theory of Operation. A four-day course for the network manager and programmer/analyst who needs to know the internal functioning of the DS/1000 software for the HP 1000 to HP 1000 link. This class provides the opportunity for hands-on programming of the system, exercising the system utilities, diagnostics, and troubleshooting tools. System generation and network reconfiguration are covered in detail. Prerequisite is DS/1000 User's Course (22987A).

22962B Theory of Operation for DS/1000 to HP 3000. This one-day course is intended for the Network Manager and Programmer/Analyst who needs thorough exposure to the internal functioning of the DS/1000 software as it relates to a DS/1000 to HP 3000 link. This course covers the listings, tables and flowcharts for the software enhancements for this communications option. Covered are such topics as: monitors, utilities, network configuration, 1000 driver, line protocol, troubleshooting, HP 1000 as Master/Slave to MPE. Prerequisite is DS/1000 Theory of Operation (22961A).

22980C HP-IB Interface with HP 1000. This four-day course familiarizes the student with HP-IB utilization, messages and device subroutines writing techniques. The student will receive hands-on experience with programmable instruments in both stand alone and system environments. Lab exercises may be programmed in FORTRAN or BASIC. Topics covered include: handshake protocol, device addressing/listen/talk, analyzing bus lines and system configuration. Session Monitor User's Course (22994A) OR Memory-Based RTE (22992A) is a prerequisite.

22983B HP 1000 E- & F-Series Computer Microprogramming. The purpose of this five-day course is to train the student in the theory and use of HP microprogramming hardware and software to prepare, alter and install microprograms. The student will learn CPU organization, microinstructions, block I/O and microprogrammable processor port, floating point processor interface, dynamic alternation of writable control store and the timing, analyzation and microcode of a CPU bound application program. Approximately forty percent of class time will be lab. Session Monitor User's Course (22994A) OR Memory-Based RTE (22992A) is a prerequisite.

22987A DS/1000 User. This five-day course is intended for students already familiar with RTE systems and who have completed either the 22992A or 22994A course. Ample lab time will develop competence in using the capabilities of Distributed Systems software in both the HP 1000 and HP 3000. Some of the areas covered are: network initialization, remote I/O, remote file access, remote EXEC calls, program to program calls, store and forward communications, utility calls and generation.

22990A RTE Driver Writing. This software concepts course is designed to help the student write a simple RTE driver without detailed knowledge of the operating system functions. It is intended for those students who have never written a driver, but who already possess a working knowledge of HP Assembly language or successfully completed the Assembler Course (22952B).

The Driver Writing course will cover I/O structure, interrupt system, interrupt drivers, driver structure and operation, use of DCPC by drivers and driver naming conventions.

22992A HP 1000 Memory-Based RTE System. This two-week user-oriented course does not cover internals. It is intended for students who program and/or manage HP 1000 computer systems with a RTE-M operating system. Students will be able to write programs that call all major system functions and perform system generations tailored to their own requirements. Some topics covered are: system theory and operator commands, utility programs, program development, segmenting, scheduling, resource management, generation, Batch-Spool Monitor, procedure files, floppy back-up and softkeys. Introduction to HP 1000 Computers (22951B.) is a prerequisite.

22993A IMAGE/DBMS 1000. This five-day course has approximately thirty percent lab time (exercises may be programmed in FORTRAN or BASIC). The course provides the user with sufficient information to use the 92069A IMAGE/DBMS components to perform the following: create, build, back-up and modify a Data Base, use QUERY and write programs to access a Data Base, as well as handle maintenance considerations and the RTE interface to IMAGE.

22994A HP 1000 RTE-IVB/Session Monitor User's Course. A 10-day course dealing with interactive and programmatic use of the RTE-IVB operating system, the file management system and the spooling system, including program development using the standard editor, compiler, assembler and loader. The course emphasis is on how to use RTE-IVB.

22995A HP 1000 RTE-IVB System Manager's Course. A five-day course on how to generate and update an RTE-IVB/Session Monitor operating system in an HP 1000 System environment.

22996A Advanced RTE Workshop. This 5-day workshop is designed for systems analyst/programmers who need to tune their systems for maximum performance. It will introduce the student to the internal design and operation of the RTE operating system. The attendees must have at least 6 months experience in using RTE and must have experience in using HP assembly language.

92780A HP-ATS Test Programming. A learning center concept is used during this five-day course to allow each student freedom to concentrate on topics of greatest interest and need. A strong electronic test equipment/procedures background is a must. The course is intended for test engineers who write test programs for a HP Automatic Test System. Content includes: architecture, session and test monitor, instrument programming in BASIC, UUT interfacing, switch control and test programming techniques. Session Monitor User's Course (22994A) is a prerequisite.

T2645A 2645A/2648A Terminal User/Applications Course. This 2-day course is for systems analyst/programmers in using the HP 2645A Display Station or the HP 2648A Graphics Terminal to solve a broad range of application problems. The student will learn how to take advantage of the extensive capabilities of these products in both off-line and on-line modes.

HP 1000 CUSTOMER TECHNICAL TRAINING COURSE PATHS

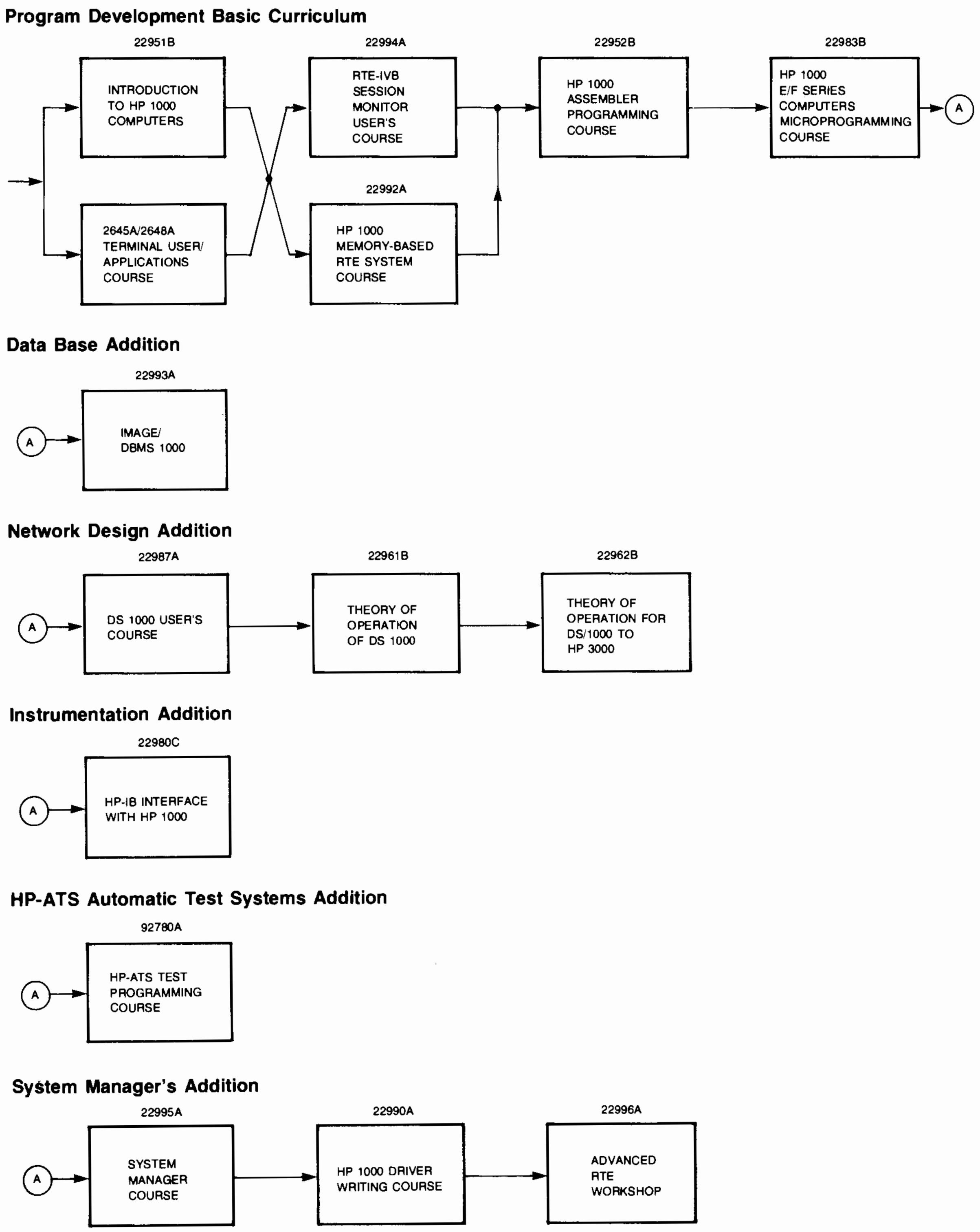

Program Development Basic Curriculum
22951B
INTRODUCTION TO HP 1000 COMPUTERS
2645A/2648A TERMINAL USER/ APPLICATIONS COURSE
22994A
RTE-IVB SESSION MONITOR USER'S COURSE
22992A
HP 1000 MEMORY-BASED RTE SYSTEM COURSE
22952B
HP 1000 ASSEMBLER PROGRAMMING COURSE
22983B
HP 1000 E/F SERIES COMPUTERS MICROPROGRAMMING COURSE
A
Data Base Addition
22993A
IMAGE/ DBMS 1000
Network Design Addition
22987A
DS 1000 USER'S COURSE
22961B
THEORY OF OPERATION OF DS 1000
22962B
THEORY OF OPERATION FOR DS/1000 TO HP 3000
Instrumentation Addition
22980C
HP-IB INTERFACE WITH HP 1000
HP-ATS Automatic Test Systems Addition
92780A
HP-ATS TEST PROGRAMMING COURSE
System Manager's Addition
22995A
SYSTEM MANAGER COURSE
22990A
HP 1000 DRIVER WRITING COURSE
22996A
ADVANCED RTE WORKSHOP

Figure 1

INTERNATIONAL CUSTOMER TRAINING SCHEDULES 79/80

	AUSTRIA	AUSTRALIA	BELGIUM	ENGLAND	FINLAND	FRANCE	GERMANY	ITALY	JAPAN	NETHERLANDS	SPAIN	SWEDEN	SWITZERLAND
22941A 21MX/XE Maint. 5 days/$500						Oct 22 (G)							
22943A 7970B/E Maint. 5 days/$500													
22945A 7905/06 Maint. 5 days/$500						Oct 29 (G)							
22951B Intro to HP 1000 4 days/$400	Sep 03			Sept 19 (W)			Sept 24	Sept 10 (R) Oct 22 (M)		Aug 27 Nov 19 Feb 11 May 05		Oct 08	
22951B-H01 FORTRAN IV 5 days							Oct 08				Oct 15		
22952B 1000 ASMB 5 days/$500	Sep 24	Oct 08 (P) Nov 12 (B)	Sept 10	Oct 29 (W)	Nov 11	Oct 08 (O)	Sep 03	Oct 15 (R) Dec 17 (M)	Jun 11 (T)	Dec 17 Apr 21	Oct 22	Sep 10 Nov 12	
22961B DS/1000 Theory of Operation 4 days/$500							Oct 01			Jan 28			
22962B DS/1000 to HP 3000 Theory of Operation 1 day/$100							Oct 05			Sep 28			
▨ 22965B RTE-II/III 10 days/$1000					Sept 10 (O) Oct 15 (O) Dec 03 (O)						Oct 29		
▨ 22969A Distributed Systems 5 days/$500						Nov 26 (O)							
▨ 22977 IMAGE 5 days/$500		Aug 20 (B) Nov 12 (P) Dec 03 (B)		Oct 15 (A)	Dec 23	Sep 24 (O)				Oct 08 Mar 17 Jun 09	Nov 12		
22980C HP-IB Interface With HP 1000 4 days/$400										Aug 20 Feb 04			
22983B 21MX/E Microprogramming 5 days/$500						Oct 01 (O)							
22984A 7920 Maint. 5 days/$5000													
▨ 22985A RTE-M 5 days/$500							Sep 10						
22987A DS/1000 User's Course 5 days/$500		Oct 22 (P)		Oct 22 (W)			Sep 17			Nov 05 Jun 23			
22990A RTE Driver Writing 3 days/$300		Aug 06 (B) Oct 01 (P) Nov 19 (B)								Oct 01 Dec 10 Apr 28			
▨ 22991A RTE-IVA 10 days/$1000	Sep 10	Sep 17 (P) Oct 22 (B)	Oct 01				Aug 20 Sept 10 Oct 01 Oct 15	Sept 24 (R) Nov 12 (M)		Sept 03 Oct 15 Nov 26 Jan 07 Feb 25 Mar 31 May 19	Oct 29	Sept 10 Oct 15 Nov 19	Sept 10

▨ Mature Product Courses

INTERNATIONAL CUSTOMER TRAINING SCHEDULES 79/80 Continued

	AUSTRIA	AUSTRALIA	BELGIUM	ENGLAND	FINLAND	FRANCE	GERMANY	ITALY	JAPAN	NETHERLANDS	SPAIN	SWEDEN	SWITZERLAND
22992A HP 1000 Memory RTE 10 days/$1000							Sept 10						
22993A IMAGE 5 days/$500													
22994A Session Monitor User 10 days/$1000					Sept 23 Nov 25								
22995A System Manager 5 days/$500													
22996A RTE-IVA-IVB Upgrade 2 days/$325													
22997A Advanced RTE 5 days/$800													
40270A Intro to HP Computers 5 days						Nov 05 (O)							
91302A 2645 Maint. 3 days/$300													

■ Mature Product Courses

INTERNATIONAL TRAINING CENTER ADDRESSES

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

AUSTRALIA
(Blackburn) B
CUSTOMER TRAINING CENTER
31-41 Joseph Street
Blackburn, Victoria, Australia
(Pymble) P
CUSTOMER TRAINING CENTER
31 Bridge Street
Pymble, New South Wales, Australia

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

ENGLAND
(Altrincham) A
Navigation Road
Altrincham
Cheshire WA14 1NU
(Winnersh) W
King Street Lane
Winnersh, Workingham
Berkshire RG11 5 AR
Tel: Workingham 784774
Cable: Hewpie London
Telex: 8471789

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

FRANCE
(Grenoble) G
5, avenue Raymond-Chanas
38320 Eybens
Tel: (76) 25-81-41
Telex: 980124
(Orsay) O
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401-Orsay
Tel: (01) 907 7825

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

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

JAPAN
(Osaka) O
Chuo Building
5-4-20 Nishinakajima
Yodogawa-Ku, Osaki-shi
Osaka, 532 Japan
Tel: 06-304-6021
Telex: 523-3624 YHP OSA
(Tokyo) T
2205 Takaido Higashi 3-chome
Suginami-Ku, Tokyo 168
Tel: 03-33-8111
Telex: 232-2024 YHP MARKET TOK

NETHERLANDS
(Amsterdam)
Van Heuven Goedhartlaan 121
Amstelveen 1134
Netherlands
Tel: 020 472021

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

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

SWITZERLAND
(Zurich)
19 Chemin Chateau — Bloc
1219 Le Lignon — Geneve
Tel: 022/96 03 22

For course prerequisites and registration information contact one of the HP training centers listed above.

Printed in U.S.A. 9/79

Part No. 5951-6111