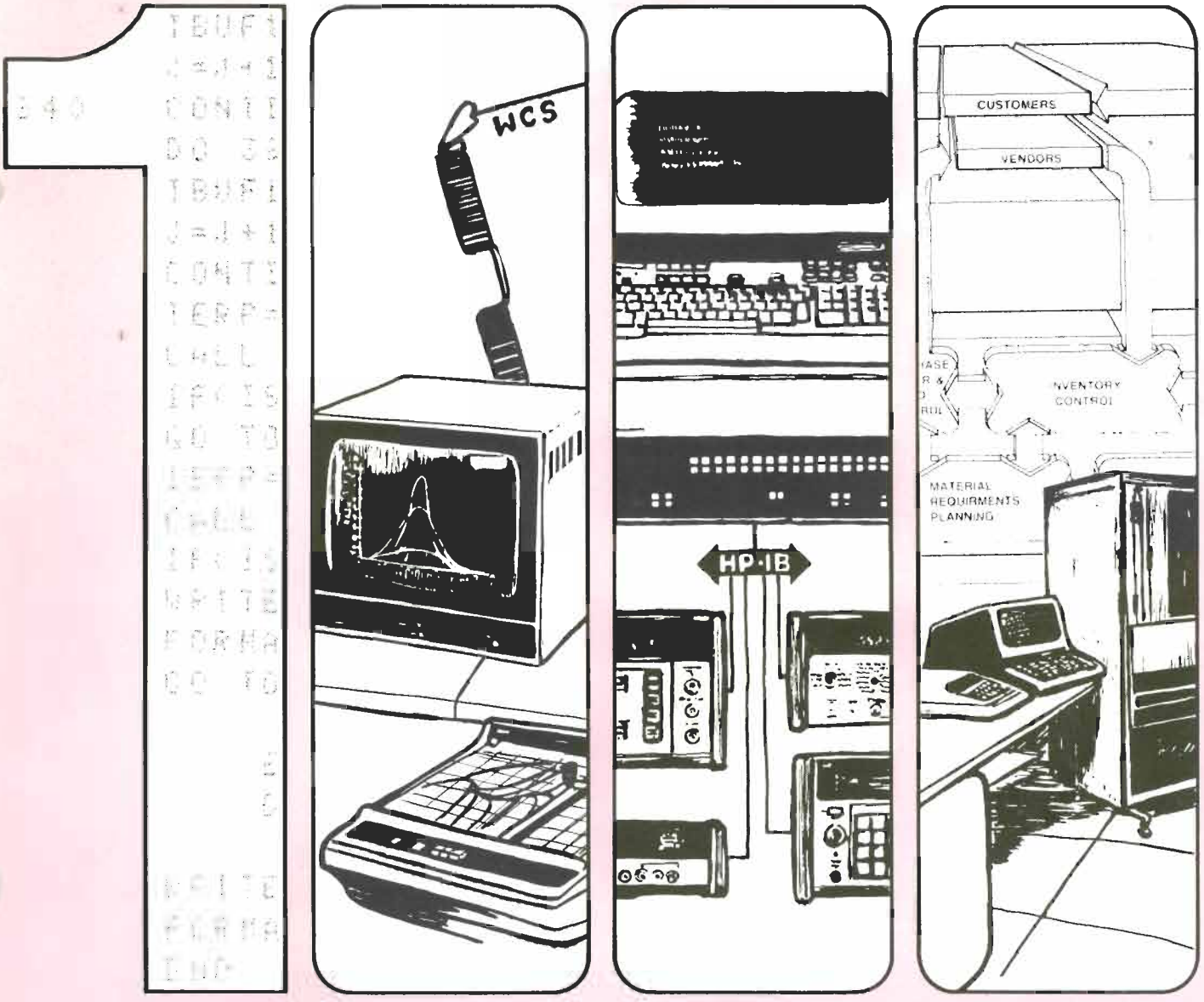
HEWLETT hp PACKARD

Computer Systems

COMMUNICATOR



1978

Volume II Issue. 2

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

EDITOR'S DESK

WIN AN HP-21 CALCULATOR!

Since its beginning in 1975, the Communicator has changed format several times. During this period, the primary source of technical articles has been employees of the HP Data Systems Division. In order to increase the diversity of topics and number of articles we are soliciting articles from customers and other HP divisions. To make it worth your time, two free HP-21 hand-held calculators will be awarded per issue (one to a customer, one to an HP employee) to the authors of the best feature-length articles which fall in one of the following categories:

Operating Systems
Instrumentation
Operations Management
Computation

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

A feature-length article must meet the following criteria:

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

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

The eligibility rules for receiving a calculator are:

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

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

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

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

All entries will be judged by a team of at least 3 people in Technical Marketing.

The number of articles currently submitted to the 1000 Communicator is not now large. If you submit an article, you have a good chance of becoming a winner!

The deadlines for articles in the remainder of 1978 are:

Issue #4 — June 15th
#5 — August 15th
#6 — October 15th

All winners will be announced in the HP 1000 Communicator in the issue in which their winning articles appear.

FOR THE OEMS

We are starting up a new section of the Communicator to be called "OEM Corner". To fill this section, we are soliciting articles HP's OEM software houses which are basically educational-tutorial in nature. To provide incentive, the OEM is permitted a short (limit 150 words) announcement of his product(s) at the end of the article which is based upon the concepts expressed in the article.

EDITOR'S DESK

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

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

Articles should be typed, double-spaced and at least 4 pages in length (exclusive of illustrations). Illustrations should be kept to a minimum. Only in unusual cases should an article be more than 10 type-written pages.

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

The deadlines for articles are the same as those listed above for the calculator prize.

CONTENTS

Starting with this issue, information on new programs in the Contributed Library (LOCUS) is given in this column. Previously, this information has been located in the "Bulletins" section.

The other purpose of this column is to publish letters from users on technical subjects. Letters may be comments on previous articles, corrections/notification of errors found, suggestions, hints for other users, etc.

NEW CONTRIBUTED PROGRAMS

This information is given to use by the LOCUS Librarian, Melanie Van Vliet. It serves as an update for the Data systems LOCUS Program Catalog (20000-90099). The programs listed below are now available. Contact your local HP sales office to order LOCUS material, or (U.S.A. only) you may use the Direct Mail Order Form at the back of this issue.

22682-18987

TAPE CARTRIDGE SAVE & RESTORE RTE-II/III

This package consists of a main program JOB3M written in FORTRAN, two FORTRAN subroutines and an Assembly language subroutine designed to operate in the RTE-II/III environment using the File Manager Package (FMP) calls and interacts with magnetic tape cartridges. This permits the user to save and restore file manager files in 128 word records, thus increasing the storage capacity of the tape to about 468 blocks. A header is written which provides a means to recreate the file manager file on recovery. A list of files is input from either keyboard or a file manager file. Any extents in the files will be packed out upon recovery. Recovery can be selected files from the cartridge or all files on the cartridge. Means are provided to copy the directory and user information.

22682-18987	PT	$25.00
22682-13387	Cass	$35.00

22682-18989

HP3437 SYSTEM VOLTMETER DEVICE SUBROUTINES

The 3437A Device subroutines are high level interface routines which convert simple call statements in FORTRAN or Basic into RTE calls to the HP-IB driver. Programming the 3437A is divided into a setup call (DSVSU) and a measurement call (DSVMS). All front panel switches of the 3437A are programmable for range, trigger-source, and delay. Up to 100 readings may be made in the direct mode and 1250 readings in a second mode where the readings are put into a disc file. Usage is made of software from the Multi-Terminal Interface Software, HP part number 92425A to be compatible in a multi-terminal environment. Minor modifications are necessary if that software is not used.

22682-18989	PT	$35.00
22682-13389	Cass	$35.00

22682-10990

ABORTING PROGRAM FREEZER —

This package allows you to fix up RTE so that it will "freeze" the program in its partition and dump the program to some device for analysis, or, better yet, use a debugging utility to determine what it was doing, without having to decode instructions. The main program is ABDMP. You run it, and specify the name of one or more programs which are to be "frozen" for your examination, should they be aborted by RTE. ABDMP inserts a "patch" into the RTE system message processor, $ERMG, so that some "special" code can be executed first. $ERMG is entered whenever RTE wants to abort a program abnormally (memory protect, dynamic mapping, IOXX errors, etc.). The "special" code

is contained in the module SPATC, supplied with ABDMP, which compares the ID segment address of the currently executing program (which is the one being aborted, usually) with the ID's of all programs in its list. The exception is when an operator aborts a program.

22682-10990	800 BPI MT	$75.00
22682-11990	1600 BPI MT	$75.00
22682-13390	Cassettes	$75.00

22682-18991

RTE-II/III MULTI-TERMINAL PROGRAM DEVELOPMENT & EXECUTION INTERFACE WITH AUTOMATIC SPOOLING

This program establishes a true multi-terminal, multi-program development and execution environment with automatic spooling. That is LS/LG bottlenecks are eliminated and any listings, reports, etc., contending for the same peripherals are automatically spooled instead of interleaved. Furthermore no cooperation is required among users that are editing, compiling loading and executing programs concurrently. The program assembles/compiles programs either from logical units and/or files. Listings and the object code may be directed to either logical units or files. If a output logical unit is a spooled device, a spool pool file is automatically set-up. The spool file is closed and passed at the end of the operation. In the case of output files, duplicate files are purged and recreated. If the file does not exist it will be created. This programs loads programs from logical unit devices and/or files and then performs a library search. Both compile and load operations may be from multiple files and/or logical units. In the case of multiple sources, tape numbers will identify each source in the listing. Standard NAMR parameters are source in the listing. Standard NAMR parameters are used for file names i.e., NAME:SC:C#. All messages are directed to the user's console instead of the system console, i.e., LOADER UNDEFINED EXTERNALS, NO SOURCE, ETC.,.

22682-18991	PT	$50.00
22682-10991	800 BPI MT	$50.00
22682-11991	1600 BPI MT	$50.00
22682-13391	Cass	$75.00

22682-10992

Status Decoding Functions

These four subprograms test the status of a logical unit for end of file, end of medium, and error conditions. They can be called as LOGICAL functions.

22682-10992	800 BPI MT	$40.00
22682-11992	1600 BPI MT	$40.00
22682-13392	Cassette	$40.00

22682-10993

WOLF - Automatic Typing Program

This is a FORTRAN IV program for the HP 2100 series computer that provides for automatic typing operations and can, when employed with manufacturers text editor, provides a system to greatly facilitate preparation of reports, letters and other text. The input text and imbedded control data can perform nearly all of the functions of a typist. A few of the features available are centering, titles, footnotes, indentation, page numbering (including Roman numerals), automatic paragraphing, and two forms of tab operations. The documentation contains both user and technical description of the program.

22682-10993	800 BPI MT	$95.00
22682-11093	1600 BPI MT	$95.00
22682-13393	Cassettes	$120.00

22682-18994

Check Protect Subroutine — PROTK

This subroutine is useful in payroll and accounts payable programs where checks are computer generated. Input is eight character decimal string, and output is sixty-eight characters.

22682-18994	PT	$10.00
22682-13394	Cassette	$35.00

22682-10995

COORDINATE TRANSFORMATION PACKAGE

This package is an adaptation of the coordinate transformation routines contained in The Coast Geodetic Survey, Tech. Report #34, 8/67 — "Aerotriangulation: Transformation Of Surveying And Mapping". Revisions have been made to include the following capabilities or requirements:

1. X, Y projection and spheroid constants either contained in or generated by the program.
2. Method of selection based on a pre-defined set of codes.
3. The incorporation of the mercator projection.
4. The ability to access the coordinate transformation routines from another program.
5. Allow full transformation capability with minimum storage requirements on the application program.

In order to accomplish the desired purpose of the transformation routines, excution time has been deemed secondary. Hardware requirements include a 21MX with minimum 8K partition and at least 138 words of background common.

22682-10995	800 BPI MT	$75.00
22682-11995	1600 BPI MT	$75.00
22682-13395	Cassettes	$95.00

USER'S QUEUE

A correction to the article "A Solution to the Multi-Terminal Blues" is issue #16 came to our attention as follows:

"Dear Sir,

The article entitled "A Solution to the Multi-Terminal Blues" in issue number 16 aroused sufficient interest in myself to try out the transfer file exampled at the bottom left of page 12. This caused a little confusion until I realised a mistake in the line which assigns a value to the global −25P. I suggest a corection of the form:

```
:CA,-25:P,-39P,/,10,*,10,*,-1,+,-39P,*,400B,+,-25P
```

I hope this may be of some use to you, perhaps for documentation purposes. However, I must congratulate you generally on your issue #16. I found it very interesting reading indeed.

Yours faithfully,
Nigel Turner (Research Scientist)
British Gas Corporation
Research & Development Division
Midlands Research Station
Wharf Lane, Solihull,
West Midlands B91 2JW"

This correction has been verified and is, indeed, accurate. Thanks to Nigel Turner for pointing it out to our readers — and for the compliments!

A second letter from a reader gives a technique that should be of interest to many others:

"Enclosed is a small assembler subroutine that allows formatted output in RTE BASIC. To my knowledge, this has not been published before and it was the one thing really missing in BASIC.

It is simple to use. Merely set up the format statements in string variables with up to 62 characters. The total number of variables that can be transferred in one WRITE statement in 12.

Sincerely,
D.A. Van Den Eijkel
ESCOM----P.T&C. Dept.
P.O. Box 103
Germiston
1400
Republic of South Africa"

The listing for the subroutine is given below. It includes a sample test program (in BASIC) and the required entry for the Branch & Mnemonic Table Generator. Many thanks to D.A. Van Den Eijkel.

```
                ASMB,R,L
00000                NAM WRITE
                     EXT .ENTR,.DIO.,.RAR.,.DTA.,IFIX
                     ENT .WRIT
*
* SUBROUTINE TO ALLOW FORMATTED OUTPUT IN BASIC
*
* SAMPLE BASIC PROGRAM:
*
*      10 DIM A$(100),B(3)
*      20 A$="(12HANSWERS ARE: ,F6.1,2X,I2,3(2X,F7.2))"
*      30 READ G,H,B(1),B(2),B(3)
*      40 DATA 327,7,64,89.56,1509.5,1421.1
*      50 WRITE (15,A$,G,H,B(1),B(2),B(3))
*      60 END
*
*  UP TO 12 VARIABLES ARE ALLOWED
*  FORMAT STATEMENT MAY HAVE UP TO 62 CHARACTERS
*      (80 CHARS PER BASIC LINE)
*
*  RTETG INPUT FILE:
*  BRT,MNT,ID=C
*  WRITE(R,RA,R,R,R,R,R,R,R,R,R,R,R),OV=0,BP,VL,ENT=.WRIT,FIL=%WRITE
*
*  ESCOM--P.T.&C.---GERMISTON-- REP. OF SOUTH AFRICA
*  D.A. VAN DEN EIJKEL
*
00000 000000  LU#   NOP
00001 000000  FMT   NOP
00002 000000  PARAM BSS 12
00016 000000  .WRIT NOP
00017 016001X       JSB .ENTR
00020 000000R       DEF LU#
00021 104200        DLD LU#,I
00022 100000R
00023 016005X       JSB IFIX
00024 072067R       STA LU        LIST LU
00025 062001R       LDA FMT
00026 002004        INA           ADVANCE PAST STRING COUNT
00027 072126R       STA IFMT
00030 062127R       LDA =D-12
00031 072070R       STA PCNT      TOTAL # OF PARAMETERS
00032 062071R       LDA DPRAM
00033 072072R       STA IPRAM
00034 006400        CLB
00035 076073R       STB PMCNT     PARAM COUNT
00036 062074R       LDA IARR1
00037 072075R       STA IARR2
*
* GET SPECIFIED NUMB OF PARAMS AND STORE IN ARRAY
*
00040 162072R LOOP  LDA IPRAM,I
00041 002003        SZA,RSS
00042 026055R       JMP ON
00043 104200        DLD A,I
00044 100000
00045 104400        DST IARR2,I
00046 100075R
00047 036073R       ISZ PMCNT
00050 036072R       ISZ IPRAM
00051 036075R       ISZ IARR2
00052 036075R       ISZ IARR2
00053 036070R       ISZ PCNT
00054 026040R       JMP LOOP
*
00055 062067R ON    LDA LU
00056 006400        CLB
00057 016002X       JSB .DIO.
00060 100126R       DEF IFMT,I
00061 000066R       DEF EOL
00062 062073R       LDA PMCNT
00063 066074R       LDB IARR1
00064 016003X       JSB .RAR.
00065 016004X       JSB .DTA.
00066 126016R EOL   JMP .WRIT,I
*
00067 000000  LU    NOP
00000         A     EQU 0
00070 000000  PCNT  NOP
00071 000002R DPRAM DEF PARAM
00072 000000  IPRAM NOP
00073 000000  PMCNT NOP
00074 000076R IARR1 DEF ARRAY
00075 000000  IARR2 NOP
00076 000000  ARRAY BSS 24
00126 000000  IFMT  NOP
00127 177764
                    END
```

Still another letter comes from John Gwyther of the Melbourne (Australia) Eye and Ear Hospital.

"Have you ever found 6 character file names a restriction for identification purposes? For example, if each user makes a source file name begin with "&" and then puts his initials in character positions 2 and 3, it leaves only 3 characters to construct a meaningful name. The spool procedure below replaces security codes user names, thus identifying the file owners and initiating a method for purging unknown files.

The system manager should assign users their individual codes (could be their initials). The only system requirement is the presence of spooling. To use the procedure.

```
*RU,JOB,DLIST
```

where the file DLIST is given below. A sample output follows the listing of the DLIST file."

```
:JOB,DLIST
:SV,2,,IH
:** THIS JOB WILL PRODUCE A DIRECTORY LISTING WITH SECURITY CODES
:** REPLACED BY USER'S NAMES.
:** FIRST SET UP A SPOOL FILE FOR LU 6.
:LL,6
:PU,DIRLST:RT:-3
:CR,DIRLST:RT:-3:4:48
:LU,6,DIRLST:RT:-3,WRST
:** PUT OUT SOME HEADER INFORMATION
:AN,***
:AN,*** THE FOLLOWING DIRECTORY LIST CONTAINS USER NAMES RATHER THAN
:AN,*** SECURITY CODES.
:AN,*** PLEASE CHECK THROUGH THE LIST AND PURGE ANY OF YOUR FILES
:AN,*** WHICH YOU NO LONGER REQUIRE.
:AN,***
:**  NOW OUTPUT SOME DIRECTORY LISTS.
:DL,-46,AA
:** CLOSE THE SPOOL
:CS,6,EN
:** RUN THE EDITR IN BATCH MODE - COMMANDS REALLY COME FROM THIS
:** PROCEDURE FILE, AND WILL MODIFY THE COMPLETED SPOOL FILE.
:RU,EDITR,5
DIRLST:RT:-3
W21,29
Z21076/SYSTEM
1
Z19010/JIM B.
1
Z16716/AL LIU
1
Z21574/TODD F
1
Z16708/AUDREY
1
Z17738/EIKO J
1
Z19789/MIKE/M
1
Z00000/??????
1
ER
:** LIST THE NEW DIRECTORY FILE
:LI,DIRLST
:EOJ,DLIST
```

```
***
*** THE FOLLOWING DIRECTORY LIST CONTAINS USER NAMES RATHER THAN
*** SECURITY CODES.
*** PLEASE CHECK THROUGH THE LIST AND PURGE ANY OF YOUR FILES
*** WHICH YOU NO LONGER REQUIRE.
***
 CR=00460
  ILAB=LU46   NXTR=0071 NXSEC=084 #SEC/TR=096 LAST TR= 0202 #DR TR=02

 NAME    TYPE #BLKS/LU SCODE TRACK SEC  OPEN TO

 BATCH   00004 00002   ??????   0000 000
 &ANSWR  00003 00028   ??????   0000 006
 &MAKE   00004 00016   ??????   0002 078
 &PAGE   00004 00005   ??????   0003 014
 &D.01   00004 00045   ??????   0003 024
 &D.02   00004 00040   ??????   0004 018
 &IOCB   00004 00086   ??????   0005 002
 &D.00   00004 00070   ??????   0006 078
 &LOADA  00004 00080   ??????   0008 026
 &LOADB  00004 00080   ??????   0009 090
 &LOADC  00004 00080   ??????   0011 058
 &LOADD  00004 00078   ??????   0013 026
 &IOC"   00004 00053   ??????   0014 086
 &IOCB'  00004 00127   ??????   0016 000
 &PCS    00004 00402   ??????   0018 062
 &D.23   00004 00147   ??????   0033 058
 COMND   00004 00001   ??????   0036 064
 &#TAM   00004 00055   ??????   0045 076
 &L65    00004 00033   ??????   0049 014
 %.IOC.  00005 00004   ??????   0049 080
 %#TAM   00005 00006   ??????   0049 088
 %TEXEC  00005 00023   ??????   0050 004
 %RFAIN  00005 00013   ??????   0050 050
 %D.65   00005 00016   ??????   0050 076
 %D.00D  00005 00007   ??????   0051 012
 %L65    00005 00004   ??????   0051 026
 $YSLB   00005 00001   ??????   0051 034
 $ALRN   00005 00002   ??????   0051 036
 RNRQ    00005 00003   ??????   0051 040
 LURQ    00005 00004   ??????   0051 046
 PRTN    00005 00002   ??????   0051 054
 EQLU    00005 00002   ??????   0051 058
 .DRCT   00005 00001   ??????   0051 062
 COR.A   00005 00001   ??????   0051 064
 KCVT    00005 00001   ??????   0051 066
 CNUMD   00005 00001   ??????   0051 068
 CNUMO   00005 00001   ??????   0051 070
 INPRS   00005 00002   ??????   0051 072
 "DFINE  00004 00104   MIKE M   0051 076
 WELCOM  00004 00001   SYSTEM    0053 092
 &D.00D  00004 00051   AL LIU    0053 094
 &.IOC.  00004 00055   AL LIU    0055 004
 &TEXEC  00004 00196   EIKO J    0056 018
 &RFAIN  00004 00098   AUDREY    0060 026
 &LOADR  00004 00316   TODD F    0062 030
 &D.65   00004 00143   JIM B.    0068 086
```

SPECIAL ANNOUNCEMENT

New Software Support Program Now Available for HP 1000 System Users

by George Taylor

A new software support program which covers its HP 1000 Computers and Computer Systems is now in effect. This program provides for six catagories of software as well as defining three levels of available support services. This program will enable customers to obtain software and manual changes resulting from product enhancements and updates. By purchasing the highest level of service a spedific telephone "hot line" is provided for telephone access to the Phone-In-Consulting Service. This contact will provide a link to a Systems Enginer to allow a maximum of four (4) hours response regarding questions on the use of **Hewlett-Packard** software.

SOFTWARE CATEGORIES — Each software product available for HP 1000 system users has been assigned to one of six categories to provide a basis for defined software support services. The ACTIVE category defines software which **Hewlett-Packard** intends to enhance. Also, all discrepancies found in software within in this category will be resolved. Software in the MATURE category will not be enhanced but descrepancies will be resolved. A COPIED category results from a customer reproduction of either Active or mature software as a result of having purchased the right-to-reproduce product. The OBSOLESCENT software category defines software which is no longer available except as replacement parts, and CONTRIBUTED software which will be available from the Library of Contributed User Software (LOCUS) program as in the past. The SPECIAL software category includes only that software designed for an individual customer, with no enhancements planned, but where discrepancies will be resolved.

SOFTWARE SERVICES — Three levels of software support services have been made available to software users. These levels permit the user to select only those support services which are necessary based on user expertise or experience with HP 1000 Systems.

The SOFTWARE NOTIFICATION SERVICE has been designed for the self sufficient customer who only wants to monitor software product activity but is not interested in upgrading software in the near term. This service includes six issues of the Communicator 24 issues of Software Status Bulletins that give corrections for reported descrepencies in software and manuals, and four Software Updage Notices which explain changes to current software/firmware and manuals.

The SOFTWARE SUBSCRIPTION SERVICE provides revised software on the medium selected by the user, as well as technical manual changes. Operating System Software Subscription Service includes the Software Notification Service.

COMPREHENSIVE SOFTWARE SUPPORT has been designed for those users who want the highest level of assistance from Hewlett-Packard. t includes the Software Subscription Service already described, and a Phone-In Consulting Service. This service permits telephone access by your System Manager to a trained **Hewlett-Packard** Systems Engineer who can help resolve questions directly related to the use of **Hewlett-Packard** supplied software.

Your Sales Representative can provide you with ordering information for ech of the products covered by this new program. If necessary, the Systems Engineer will visit the user to help resolve questions concerning HP software.

RTE-IV — LOGICAL AND PHYSICAL MEMORY ORGANIZATION

David L. Snow

RTE-IV is Hewlett-Packard's latest disc based operating system for its HP 1000 family of computers. This is the first in a series of articles by the RTE-IV development team which will discuss various RTE-IV structures and processes. This article concentrates on the differences between memory structures in RTE-III and IV. Future articles will discuss topics such as I/O and Memory Reconfiguration, EMA Firmware Algorithms, User Partition Management, etc.

RTE-IV, announced in April 1978, is Hewlett-Packard's newest operating system for managing up to 2 Mbytes of memory in the HP 1000 series of computers. Besides supporting all the capabilities of its predecessors, RTE-II and III, RTE-IV adds several significant enhancements. These include the following:

a. Guaranteed user code area of 54 Kbytes regardless of future increases in system code area or the addition of new drivers.

b. User data areas resident in memory of up to 2 Mbytes with access transparent to the FORTRAN user.

c. Detection and removal from usage of defective pages of memory found by the operating system in the use's memory area.

d. I/O and memory reconfiguration at boot-up.

e. Improved Multi-Terminal-Monitor management with automatic creation of individual copies of user programs.

f. I/O from and to files for FORTRAN, Assembler and Loader operations.

g. A powerful assembly language level debugger routine.

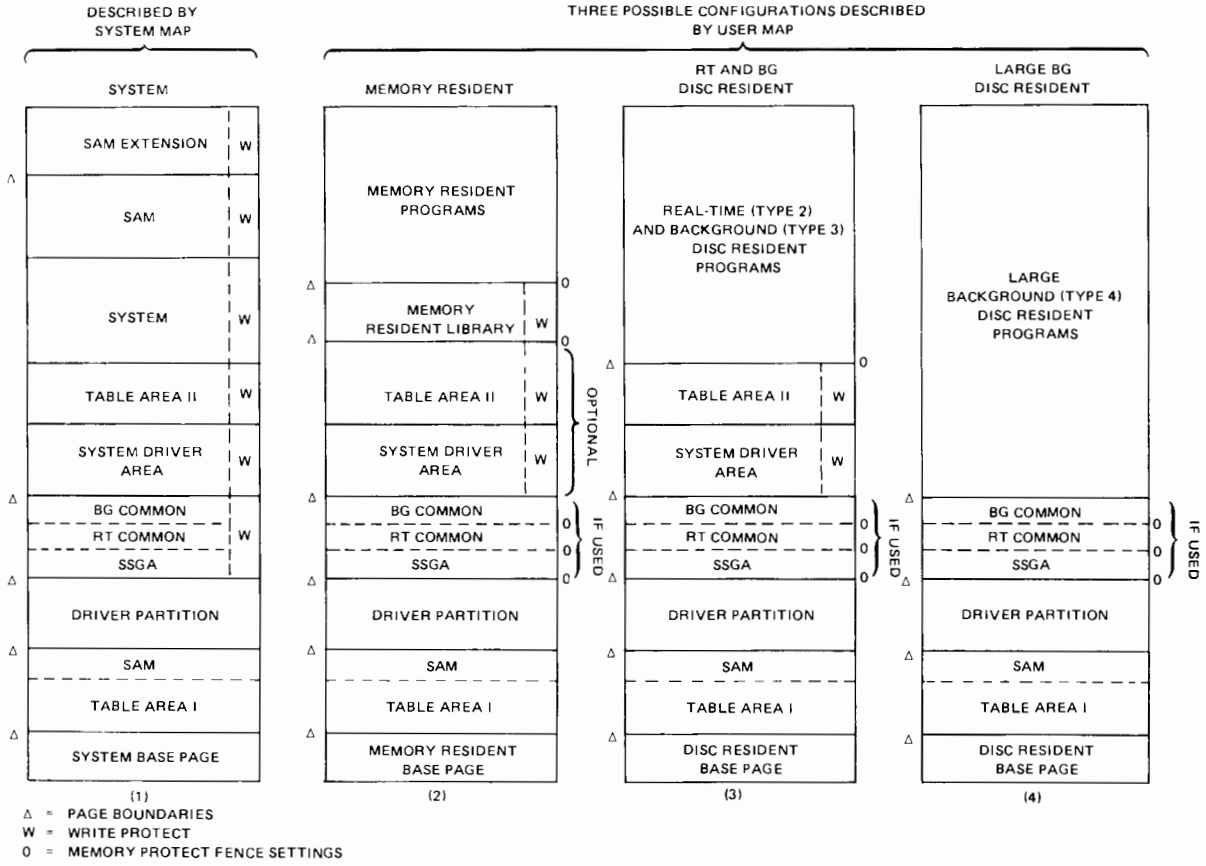


Figure 1. RTE-IV Word Logical Memory Configurations

A major difference between RTE-III and IV is in the structuring of logical and physical memory. This was necessitated by a desire to reduce to a minimum the amount of user area occupied by portions of the operating system. Figure 1 shows the layout of logical memory for the RTE-IV operating system map and for memory maps associated with type 2, 3, and 4 programs. For comparison, Figure II shows similar maps for RTE-III. Out of the operating system, only commonly used entry points (e.g., EXEC, $LIBR, $LIBX) and constants (e.g., $OPSY, $TBxx) are included in all maps. These entry points and constants along with all I/O tables (Device Reference Table, Equipment Table and Interrupt Table) are included in a new area called Table Area I. This insures that most user programs requesting Executive services and most user drives accessing data in I/O tables will execute in all RTE-IV addressing spaces without the need of recoding the process.

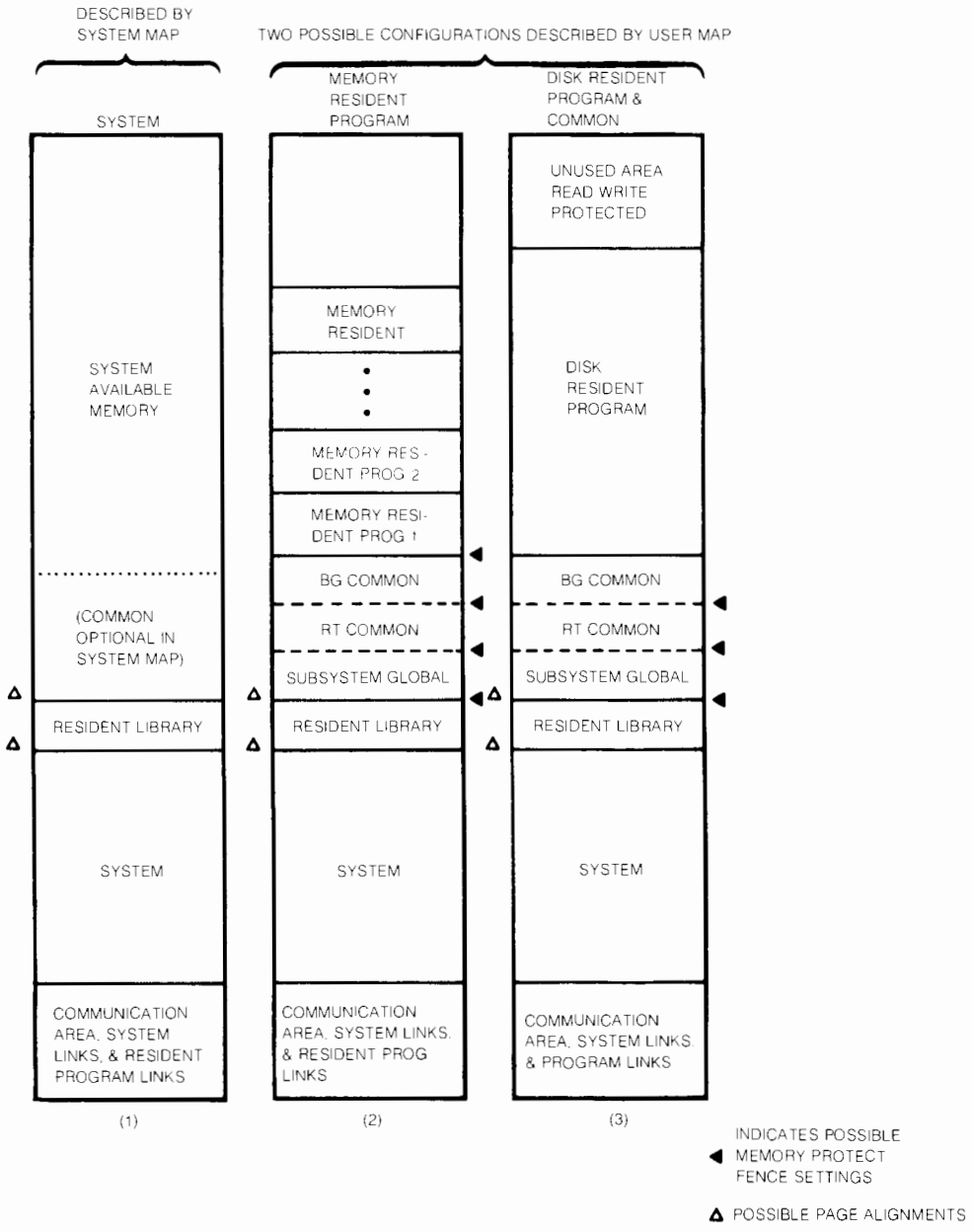


Figure 2. RTE-III Logical Memory Configurations

A second area of memory normally two pages in length is included in all addressing spaces and called the driver partition area. At generation time, all drivers are grouped via a best fit algorithm into two page partitions. In physical memory these partitions are placed immediately following system-available-memory (see Figure III). New Driver Mapping Tables, one associated with each Equipment Table entry, indicates which driver is contained in which driver partition. Whenever an I/O request is processed, the appropriate driver partition is mapped into the user or system addressing space prior to the driver's execution. This insures that only the driver actually in use is included in the appropriate addressing space.

The reduction in system size made possible by Table Area I and the reduction in driver area made possible by driver partitions, allows RTE-IV to specify a guaranteed user coding area of 54 Kbytes for type 4 programs.

Another area of memory called Table Area II which contains lesser used system entry points and constants (e.g., $MATA, $TIME, $BATM) and other system tables (e.g., ID segment table, class number table, resource number table) was created and included in the addressing space of the system and type 2, 3, and, optionally, type 1 programs. This area insures that all previously written user programs for RTE-III which access these areas will execute properly.

A second driver area, the System Driver Area, is included in the addressing space of the system, type 2, 3 and, optionally, type 1 programs. This area was created for privileged drivers (which must always be in the system map), drivers which do their own mapping and drivers larger than the driver partition size (usually two pages).

With the System Driver Area, Table Area II, Driver partition Area and Table Area I, the addressing space of type 2 and 3 programs will typically be 40 Kbytes, up from the 28 to 32 Kbytes associated with RTE-III.

Within the system map, RTE-IV provides three areas of memory which make up system-available-memory (SAM). At boot up, the first three pages of physical memory following the system area are used for I/O and memory configuration. After bootup, this area is allocated to SAM. The user can at generation optionally declare a SAM extension. This area, although contiguous to the first SAM in the system's logical memory, is actually located in physical memory between the Memory Resident Program area and the User Partitions. In addition, the unused portion of the last page occupied by Table Area I is included in SAM.

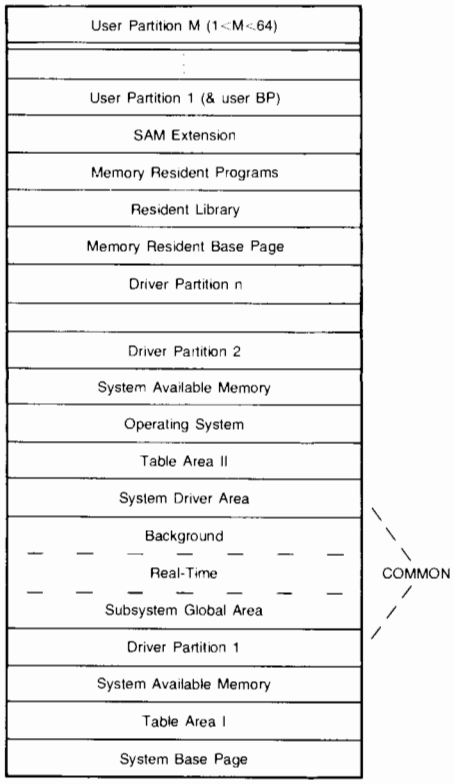


Figure 3. Physical Memory Allocations

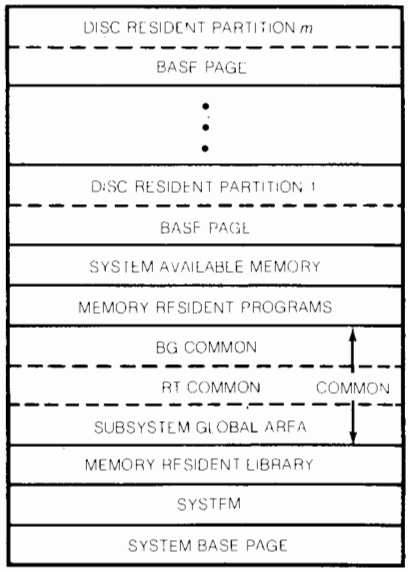


Figure 4. RTE-III Physical Memory Allocations

Figure III shows the physical memory layout of RTE-IV. For comparison Figure IV shows the layout for RTE-III. The major differences are the positioning of Common before the system, the positioning of drivers in driver partitions and the creation of the SAM extension after the memory resident area.

In conclusion, RTE-IV has restructured the logical memory maps associated with the system and user areas in order to maximize the user's addressing space. Next month's article will discuss the management of data areas of up to 2 Mbytes by firmware routines.

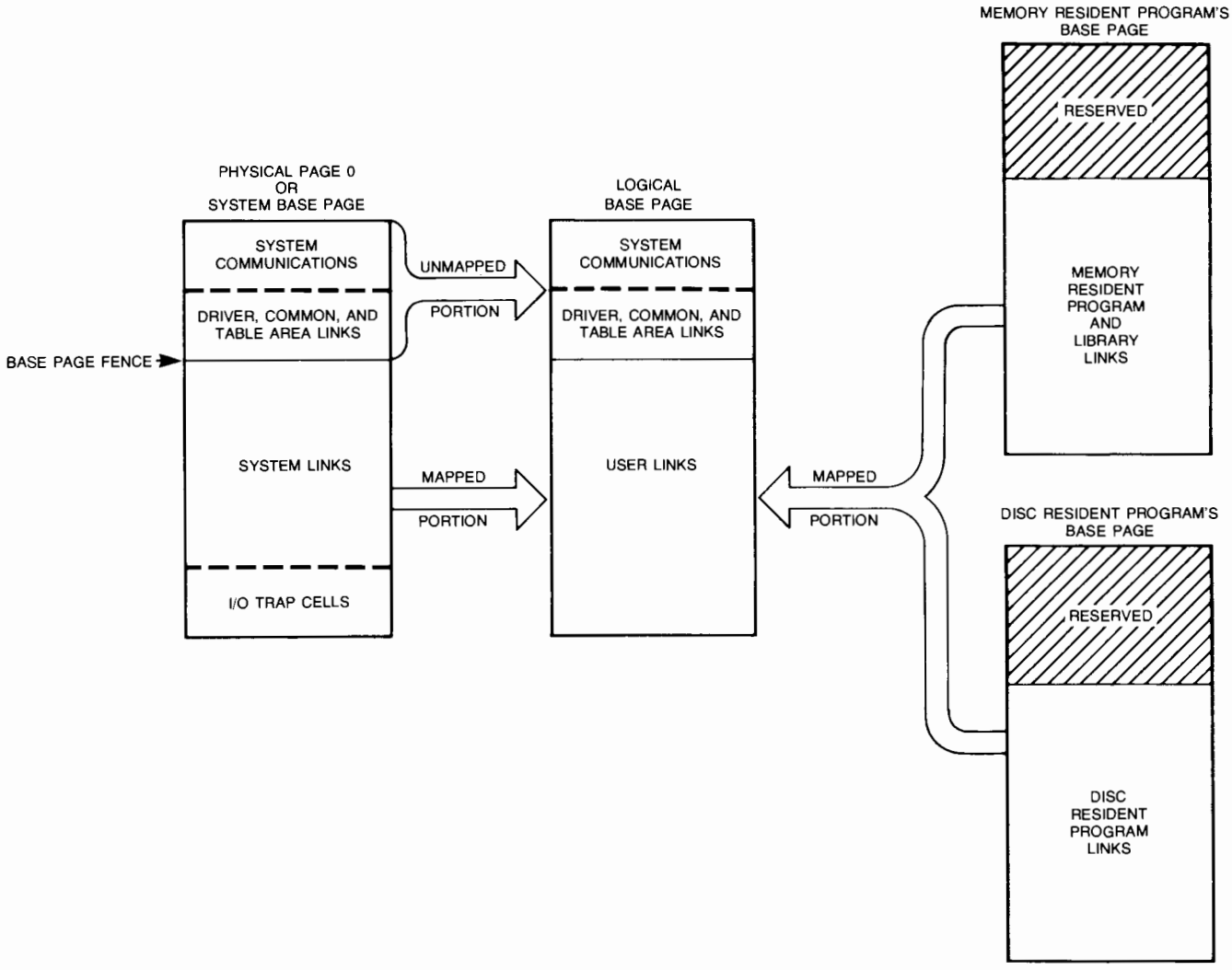


Figure 5. System and User Base Pages

KNOW YOUR DS/1000, PART I

Al Liu/DSD

This is the first of a series of articles on the internals of the DS/1000 software. To comprehend these articles requires a basic understanding and knowledge of DS/1000 which is described in the DS/1000 Programmer's Reference Manual and the DS/1000 Network Manager's Manual.

The intention of this article is to provide an introduction to the internals of DS/1000 by giving:

A. a description of the layered architecture of DS/1000 software modules.

B. a description of the transaction flow from a DS/1000 source node to a DS/1000 destination node by tracing a CALL DEXEC (2) as an example.

NOTE!!!

DS/1000 software is supported by HP only at the user interface level documented in the DS/1000 Programmer's Reference Manual. Subroutines not documented which are described or mentioned herein or elsewhere in the series, must never be called directly from a user program. They are not supported by HP when used in such manner. They are described or mentioned solely for the purpose of illustration. Their functions and calling sequences are subject to change without notice to any user.

The second article in this series will describe:

a. the lists and the nodal entry points in the resident subsystem global area (SSGA) module called RES.

b. a node's initialization by LSTEN.

A. THE LAYERED ARCHITECTURE OF DS/1000

[The following diagram and some of the terminologies are from pages 4 and 5 in the "Distributed Systems/1000 Technical Data" brochure (part number 5953-0868).]

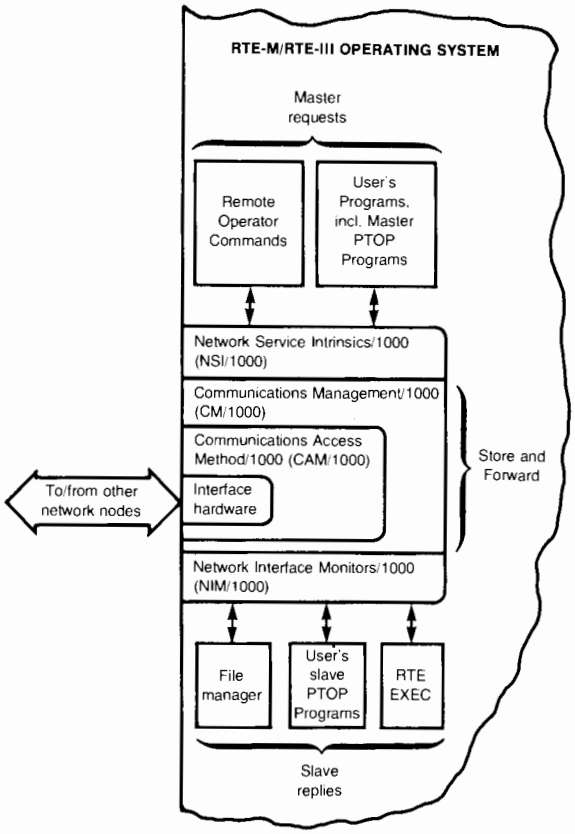
DS/1000 is composed of layers of software modules. Each module is designed to perform a special task (i.e., task-oriented) for a certain level of processing of a transaction. Modules processing at the same conceptual level on both sides of a link are classified into a layer.

The bases for this architectural design are:

1. To minimize the occurrence of any "bottleneck" condition in communicating with multiple users and/or nodes in a network. Most of the modules are serially re-usable, each module for a task at a time. By the fact that each module is dedicated to perform a specific task, its processing is minimized so that it will be available as soon as possible for the next caller.

2. To allow new kinds of link to be added in the future without affecting user's program codes.

The DS/1000 software modules are "layered" as in the following diagram.



The Network Service Intrinsics/1000 (NSI/1000) Layer

The Network Service Intrinsics (NSI/1000) are:

1. called from user's programs or by operator commands in REMAT and RMOTE,
2. to generate the transaction format for master requests to remote nodes, with data as required.

The user program calls are in the forms of DEXEC, PTOP, RFA and some remote utility calls. The NSI/1000 modules are type 7 subroutines (i.e., utility subroutines) which are appended to the user program that calls them.

The results from executing each of these subroutines are:

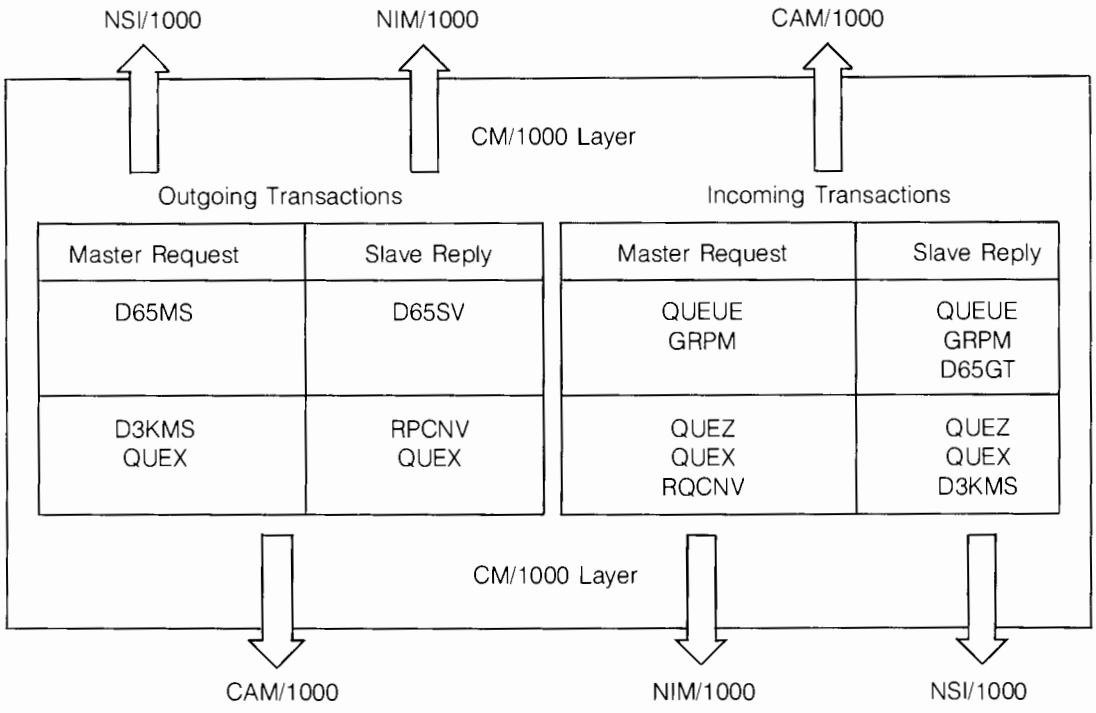
- A "request buffer" is set up for the respective request according to the formats specified in Appendix A in the "DS/1000 Network Manager's Manual" (part number 91740-90003).
- The request buffer is then passed down to the Communications Management (CM/1000) layer as a parameter in a direct call to the appropriate CM/1000 module.
- A data buffer may or may not accompany the request buffer, depending on the nature of the request. For example, CALL DEXEC (2) for remote write is accompanied by the data buffer that is to be sent. CALL DEXEC (1) for remote read does not have a data buffer sent with its request buffer.
- The request buffer initially is local to the NSI/1000 subroutine and the data buffer, if any, resides in the calling user program. However, when they are passed to the CM/1000 layer, they are copied to a class buffer in system available memory (SAM) as the result of a class write-read call.

The Communications Management/1000 (CM/1000) Layer

This is the heart of communication processing through which all outgoing and incoming transactions must pass.

- The outgoing transactions originate either from the NSI/1000 layer if they are master requests, or from the Network Interface Monitors/1000 (NIM/1000) layer if they are slave replies. They are routed to the drivers (DVA65 or DVG67) in the Communications Access Method/1000 (CAM/1000) layer for outputs.
- All incoming transactions enter the system via the CAM/1000 layer. They are routed to the NSI/1000 layer if they are slave replies to complete the original master requests or to the NIM/1000 layer if they are new master requests received.

The following table presents the names of the modules in CM/1000 layer and their positions in the transaction flow.



The CM/1000 also manages all store-and-forward operations. For example in a DS/1000 to DS/1000 communication link, when GRPM detects that an incoming transaction is not destined for the local node, it will "forward" that transaction according to its nodal routing vector table to the next node. It will immediately route that transaction as an outgoing transaction to DVA65 in the CAM/1000 layer to be sent out.

The Communications Access Method/1000 (CAM/1000) Layer

The modules in this layer are the software and firmware drivers:

The driver for the DS/1000 link is DVA65 and the microcoded firmware.

For the DS/3000 link, DVG67 is the physical driver, HSLC is the logical driver and D$EQT is the EQT extension for DVG67.

CAM/1000 provides a line protocol for the control of communications input and output, including error checking and corrections by retransmissions at the line level. Here it provides the first level of error retries. A transaction can be retransmitted up to 7 times before exiting the driver to go back to the CM/1000 layer for the second level of error retry.

The Network Interface Monitors/1000 (NIM/1000) Layer

These are the software modules that receive their respective incoming master requests destined for the local node from the CM/1000 layer and process them. They are also called slave monitors because they are activated and function according to the master requests directed to them from remote nodes.

Processing means linking a master request to one of the following:

- RTE EXEC if they are DEXEC requests (and remote utility calls),
- FMP if they are RFA requests,
- user programs which are slaves to the remote master programs in program-to-program communication if they are PTOP requests.

When RTE EXEC, FMP or the user programs complete their functions requested of them, their results are passed back to the respective NIM/1000 modules which in turn send the results as outgoing slave replies back to the source node.

The NIM/1000 modules are:

EXECM	DLIST
EXECW	PROGL
RFAM	OPERM
PTOPM	CNSLM

Two other monitors do not send slave replies. They are used in error conditions only. QCLM prints error messages for GRPM, which is a CM/1000 module, and RTRY aids the second level error retries of outgoing transactions for GRPM.

B. THE TRANSACTION FLOW BETWEEN DS/1000 NODES

A transaction which initiates an interaction with a remote node is known as a MASTER REQUEST. The master request is not completed until its SLAVE REPLY from the destination node is received without error back at the source node. (It may also be "completed" when it times out.) Therefore, any outgoing master request expects an incoming slave reply at a deferred time in order for it to complete. Similarly, an outgoing slave reply is the response to an earlier incoming master request to complete that request at the remote source node.

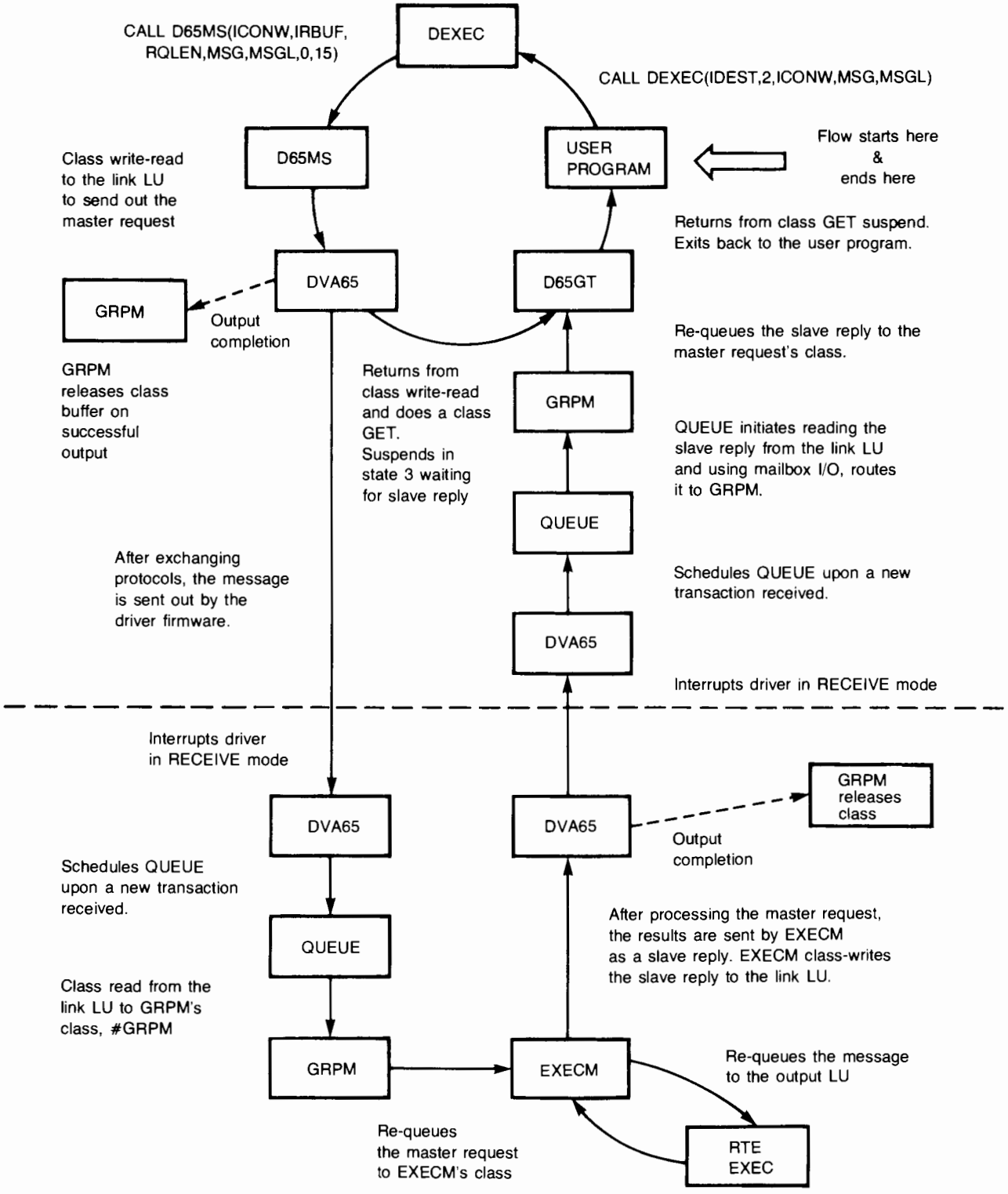
Let us trace the CALL DEXEC (2) for remote write from a DS/1000 node to another DS/1000 node in order to describe in detail the transaction flow. The following diagram is a pictorial representation of this flow. By the side of each flow path is an explanation of how the transaction is being passed to the next module.

In the descriptions that follow, you will find frequent mentions of requeueing a transaction from a class to another class or to an EQT. This is a clever scheme designed into DS/1000 software to conserve the usage of SAM (system available memory) and to minimize the system's overhead while a transaction is passed from one module to another. Once SAM has been granted as the class buffer for a transaction when the transaction is initially formed, that class buffer is saved and passed from module to module as it goes through the communication layers. Finally, after its contents have been sent out of the node, the class buffer is released back to the blocks of free SAM in the system.

Requeueing, done by a privileged memory resident subroutine called #REQU, simply means moving the class buffer pointer from the queue under a class number to another queue under the other class number or to an EQT's (except an EQT for a disc) queue if LU has been specified. If a transaction is re-queued to an EQT which is not active, #REQU initiates I/O operation on the EQT by calling a subroutine in RTIOC.

In this scheme, the succeeding processing modules do not request any more class buffers for the transaction, yet still can make use of the mailbox I/O method. This reduces the possibility of impeding any transaction due to lack of SAM during the intermediary transitions. It also avoids the overhead in intermediate allocations/de-allocations of class buffers and the overhead in moving words from buffer to buffer.

A PICTORIAL SUMMARY OF AN ERROR-FREE TRANSACTION FLOW [DEXEC (2) CALL]



THE ORIGINATING NODE

1. User Program

   The user program initiates this flow by:

   ```
   CALL DEXEC (IDEST,2,ICONW,MSG,MSGL)
   ```

   where

   IDEST = the destination node number to which the message is addressed
   2 = the code for "write" request
   ICONW = the control word for the write request contains the output LU, e.g., a line printer, for the message at the destination node (same as ICONW in CALL EXEC (2)).
   MSG = the message buffer to be sent
   MSGL = the length of the message (word count if positive, byte count if negative)

2. DEXEC Subroutine

   (This is a type 7 subroutine and is appended to the user program.)

   Major tasks:

   a. Sets up the request buffer for the message destined for a remote node.
   b. Calls D65MS subroutine, passing it the request buffer and the message for further processing.

The format of the request buffer is in the following diagram. The asterisk denotes the word which is filled in by DEXEC.

REQUEST

word 0	Stream type = 5	*
1	Sequence number	
2	Source node #	
3	Destination node #	*
4	ICODE = 2	*
5	ICONW	*
6	Message length	
7	(1st optional param)	
8	(2nd optional param)	

[It first distinguishes whether the message is destined for a remote node or for the local node. "IDEST" in the calling parameters is compared to the local node number specified in the nodal entry point #NODE in RES. If the message is destined for the local node, it will be routed to RTE's EXEC or REIO subroutine to be printed. If "IDEST" is −1, the call is also destined for local processing.

It calls D65MS with the following parameters:

ICONW = the control word passed from the user program
IRBUF = the address of the request buffer
RQLEN = the request buffer length in word count
MSG = the address of the message buffer to be sent
MSGL = the message length
0 = the length of data to be read, which is zero in this case because in the write request, there is no data to be read.
15 = the maximum length in word count of the slave reply to be received]

3. **D65MS Subroutine**

(This is a type 7 subroutine and is appended to the user program.)

Major tasks:

a. Allocates a class number for the master request.

b. Identifies the origination of the request by setting the local node number into the request buffer as the source node.

c. Registers the master request onto the list of master requests which have been initiated at the local node and which are currently still outstanding (i.e., unreplied). (Each master request is represented by a transaction control block, i.e., a TCB, in the programming logic.)

d. Sets a sequence number which is unique to each master request and slave reply, into the request buffer and into the associated entry in the master request list.
This sequence number plays the important role of matching the outstanding master request with its slave reply when the slave reply is received at a later time.

e. Converts the destination node number into its corresponding LU number by looking up the NRV table.

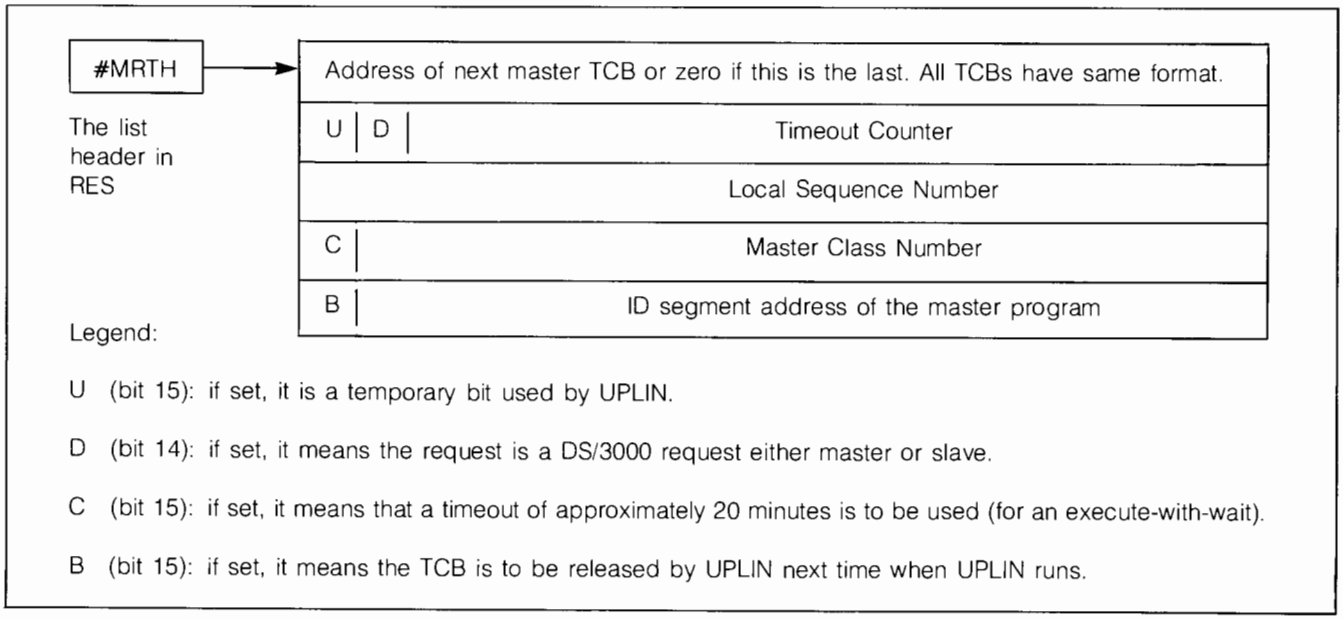
f. Sends the request and the message to the LU via the class write-read call, CALL EXEC (20). The request buffer and the request buffer length are passed as "optional parameters" in the call. GRPM's class number is specified as the last parameter in the call so that GRPM will process the I/O completion status, and institute any transmission retries, if required.

Upon returning from this class call, the request and the message buffers are copied into a SAM buffer allocated to the master request's class.

g. Calls D65GT to wait for the slave reply from the destination node. D65GT basically does a class GET on the master request's class. This causes the user program to be suspended in state 3 until one of the following occurs:

- The master request is timed out, or
- an error reject ("STOP") from the destination node has been received, or
- the proper slave reply is received.

The format of a master TCB



4. DVA65

The request and the message are routed to the driver via the class write-read call to the LU. After exchanging protocols initially for handshaking with DVA65 on the other side, the request buffer and the data buffer are output by the driver firmware as the transmission text. DVA65 regains control at the end of text and exchanges protocols to complete the transmission with DVA65 on the other side.

5. GRPM

GRPM comes out of its class GET suspension when the outgoing transmission is complete.

Major tasks:

a. If the transmission is without error, it releases the class buffer that contains the master request.

b. If there is error in transmission or if the transmission has been rejected by a "remote busy" condition at the destination node, GRPM determines if the retry count in the request has been exhausted. If retry is still possible, GRPM will re-queue the request to a NIM/1000 module called RTRY so that RTRY can route the request back to GRPM at a later time for the retry. If the request's retry count is exhausted, GRPM passes an error to QCLM, a NIM/1000 module and releases the request buffer in SAM.

c. In all cases, GRPM returns to a class GET suspension.

THE DESTINATION NODE

1. DVA65

The interface card at the destination node is enabled in the RECEIVE mode and is interrupted by the incoming request.

Major tasks for DVA65 at this point:

a. Begins an exchange of protocols in order to handshake with DVA65 at the sending node. The pending request is not yet input at this time.

b. As part of the protocol exchange, checks the status of the program QUEUE which it is about to schedule, by examining word 16 in QUEUE's ID segment. If the status indicates "not idle", DVA65 outputs "STOP" immediately to the sending node to stop further transmission. (This is a cause of "Remote Busy" condition to the originating node.)

c. If the status indicates QUEUE is idle, DVA65 schedules QUEUE by calling $LIST with QUEUE's ID segment address as a parameter.

d. It then re-enables the interface card back to the RECEIVE mode before exiting.

2. QUEUE

It is scheduled by DVA65 when a new transaction is to be read from a line.

Major tasks:

a. Performs validity checks on:
   - the interrupt source being from an initialized link
   - the data and the request lengths passed from DVA65 via the link's extended EQT

b. Initiates a class read from the interrupting link's LU to GRPM's class designated by the class number in #GRPM, which is an entry point in the subsystem global area (SSGA) initialized by LSTEN.

3. GRPM

When the class read initiated by QUEUE to read from the LU to GRPM's class is complete, GRPM will come out of its suspend state 3 from a previous class GET call.

Major tasks:

a. Registers the request as a task for the corresponding slave monitor (in this case, EXECM) to perform by adding it to the list of tasks under the slave monitor. The slave monitor is identified by the "stream type" in the request buffer. The following chart displays this association.

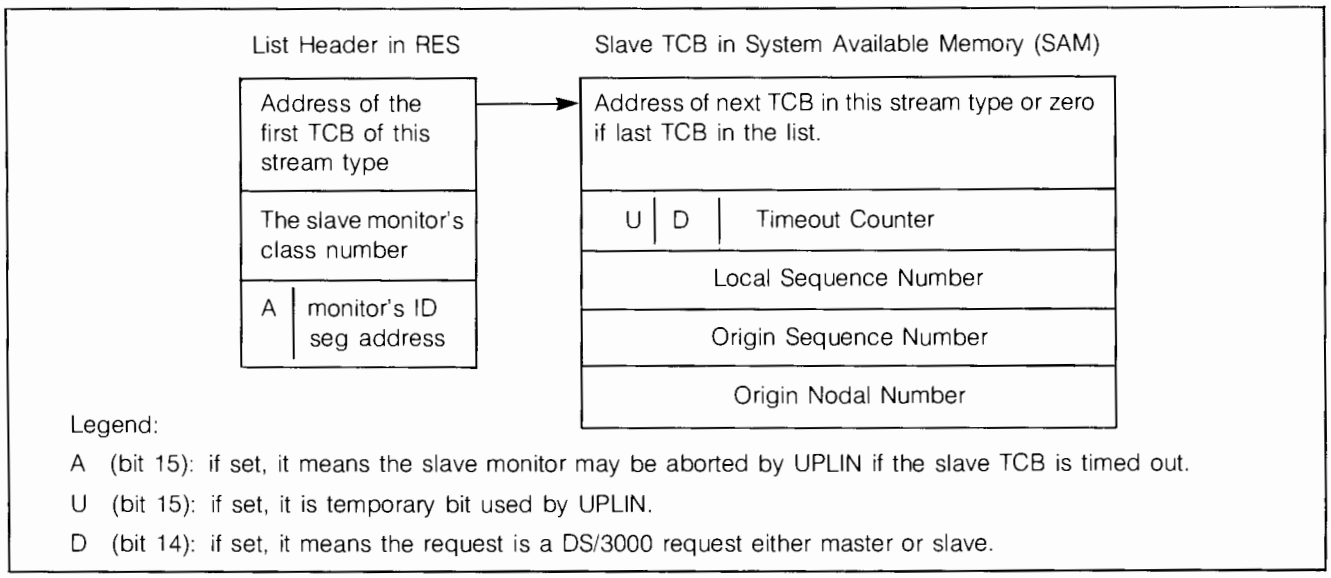
"Stream Type"	Slave Monitor
01	DLIST
02	CNSLM
03	EXECW
04	PTOPM
05	EXECM
06	RFAM
07	OPERM
09	PROGL

"Stream type" 00, 08 and 10 are reserved for future expansion.

b. Re-queues the master request in the GRPM's class to EXECM's class in order for EXECM to process this requested task.

c. GRPM returns to another class GET call and is suspended until another request is routed to its class.

The following diagram displays the format of a slave monitor's header for its list of tasks and the format of each task entry. (A task is represented by a "transaction control block" (TCB) in the programming logic.)



Legend:

A (bit 15): if set, it means the slave monitor may be aborted by UPLIN if the slave TCB is timed out.

U (bit 15): if set, it is temporary bit used by UPLIN.

D (bit 14): if set, it means the request is a DS/3000 request either master or slave.

4. **EXECM**

EXECM will come out of its class GET suspend state 3 when the request has been queued to its class by GRPM.

Major tasks:

a. EXECM determines the request, among all possible DEXEC requests, to be a "write to the specified LU at the node".

b. The received message is then re-queued from EXECM's class number to the output LU specified in the request buffer, as a class write.

c. A class GET on EXECM's class number follows in order to wait for the completion of the outputting of the message (or for a new incoming master request).

d. When EXECM comes out of its class GET suspend, it de-allocates the task (i.e., the representing slave TCB) from its list of tasks since it has completed the associated request.

e. EXECM re-uses the request buffer in SAM for its reply. Re-using this buffer is for conserving SAM usage and is the reason why EXECM handles its own slave reply whereas other NIM/1000 modules call D65SV for theirs. The slave reply is set up into the re-used buffer in SAM according to the following format.

REPLY

word 0		octal 40005
1		Sequence number
2		Source node #
3		Destination node #
4		ECOD1
5		ECOD2
6	A	ECOD3

where

A = 0 indicates normal completion:
ECOD1 = status in EQT5
ECOD2 = transmission log
ECOD3 = node number where the reply originates

A = 1 indicates error condition:
ECOD1 and ECOD2 contain 4-character ASCII message (as do A and B registers).
ECOD3 = the node number where the error occurred.

f. EXECM looks up the link LU for the originating node from the NRV table. Then it re-queue's the slave reply to the link LU to be sent on GRPM's class. (Here, DVA65 will be activated for the outgoing reply transmission. Upon the output completion, GRPM will be notified.)

g. EXECM goes back to its class GET and is suspended waiting for the next DEXEC request.

5. **GRPM**

When the slave reply has been completely output by DVA65, GRPM will be brought out of its class GET suspend. After it has determined that a slave reply has been sent without errors, it releases the class buffer which has been allocated to the original incoming master request.

GRPM returns to its class GET suspend to wait for next transaction.

THE WRAP-UP AT THE ORIGINATING NODE

1. **DVA65 and QUEUE**

When the slave reply is received at the originating node, it is processed by these two modules in a manner similar to the descriptions given for the destination node.

2. **GRPM**

When GRPM receives the incoming slave reply without any error, it searches the master request (represented by TCB) list for one with the same "sequence number" and gets the class number assigned to the master request (specified in the master TCB). GRPM, then, re-queues the slave reply to this master request class.

3. **D65GT and D65MS**

The user program has been in suspend state 3, resulting from D65GT's class GET call on the master request's class number. When GRPM re-queues the slave reply to this master request class, D65GT regains control and then passes the control to D65MS when it exits.

D65MS determines that this is a good return (i.e., not time-out nor error return). It then does the final wrap-up of the transaction flow by:

- clearing all pending writes or reads queued on the master request class,
- releasing the master request class buffer and class number
- removing the master request by de-allocating the master request TCB from the master TCB list, #MRTH.

D65MS, finally, exits back to DEXEC, which in turn, exits back to the user program, thus completing the transaction flow in a full circle.

USING CLASS I/O FOR A TERMINAL HANDLER

Gary McCarney/Rockville

Introduction

class I/O is one method of performing input and output operations in an RTE system. A program initiating a class I/O transfer does not wait for the I/O to complete. Using class I/O, many transfers can be started by a single program without waiting for the first transfer to complete. Later, using an appropriate request, the same program or another program can complete the operation by picking up the data or just checking status. Most useful applications involve one program to initiate and one or more different programs to complete the I/O. However, this article will develop an application where only a single program is involved.

This intent of this article is to develop the need and usage of class I/O by using a terminal handler example. Such a handler might serve to control how much access a user may have to the various RTE functions and utilities. This presupposes that the multi-terminal monitor (MTM) provides too little control between the intended user and the operating system. Various input methods will be discussed and the need and uses of class I/O will be developed throughout the article.

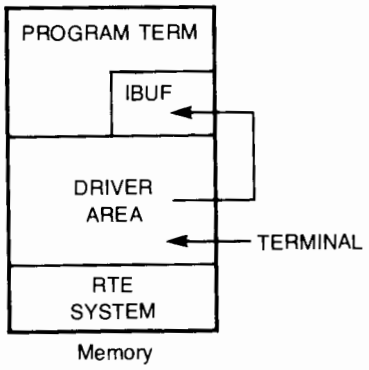
Class I/O can be used for reading, writing and control requests. This article will discuss only the use of class I/O read requests since this is the more useful aspect for a terminal handler. Most output devices utilize the automatic output buffering feature of RTE and programs do not wait for the device to complete the output transfer.

The reader is assumed to have attended one of the RTE training classes or to have the equivalent background.

Single Terminal System

When there is only a single terminal to be controlled by a user program, a dialogue could be established by using an EXEC call, an REIO call or a FORTRAN READ statement. (A FORTRAN READ will, internally, generate a REIO call.)

Let's assume throughout that the program sends some message to the user indicating the input expected and then issues a read request to the terminal. If an EXEC or REIO call is used for the input, the program will stop executing until the input is complete. The driver will be set-up by RTE to transfer the input from the terminal directly to the program's buffer (see Figure 1). The executive request used is a read (EXEC 1) from the terminal (LU) with echo enabled (400B) so that the user can see what is typed. The ASCII input data will be stored into array IBUF with a maximum of 36 words to be accepted.




```
PROGRAM TERM
DIMENSION IBUF(36)
      .
      .
      .
CALL EXEC(1,LU+400B,IBUF,36)
      .
      .
      .
```

Figure 1. Input Using an EXEC Request

While the user is typing, program TERM goes into an I/O suspended state until the input has been completed (i.e., user types RETURN). If the EXEC call is made, then while TERM is suspended, it is locked into memory and cannot be swapped because the input is being written directly into the program's space.

Since swapping is normally desired in RTE, the fact that TERM is locked into memory is not acceptable. However, if the EXEC call is changed to a REIO call, RTE will allocate an area in system available memory (SAM) that is the same size as the input buffer (plus 7 words overhead). Then the data transfer will go into SAM until the input is complete. In either case (EXEC call or REIO call) the program waits for the I/O to complete but while using REIO it can be swapped. On completion, RTE transfers the input buffer from SAM to the program's buffer and returns the SAM buffer for other use (see Figure 2). Note that the only change required in the program is to change the letters EXEC to REIO.

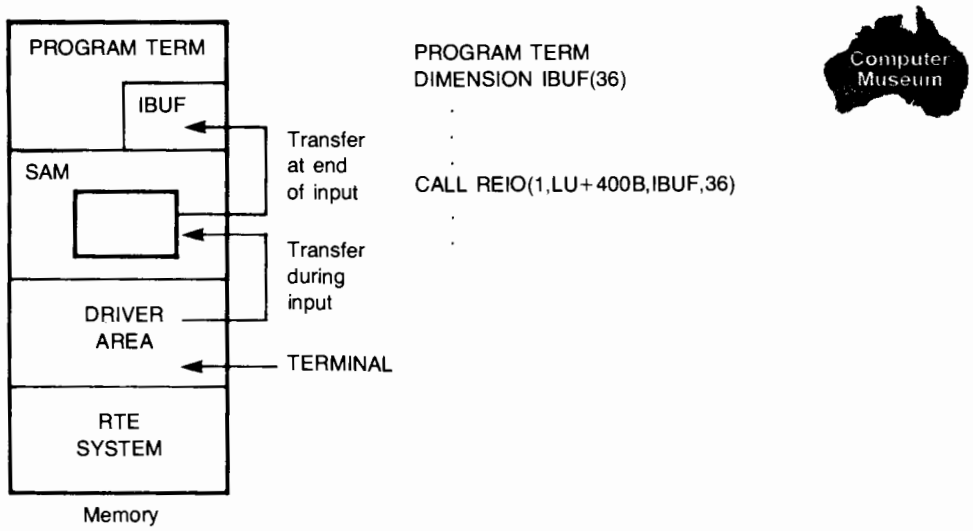


Figure 2. Input Using An REIO Request

Multiple Terminal System

A problem exists if a second terminal to be controlled by program TERM is added to the RTE system. Program TERM can input a record from only one terminal at a time since TERM goes into a suspended state waiting for the input to complete. If it is desired for both terminals to be communicating simulateneously, two versions of TERM are required — TERM1 and TERM2. Both versions are identical except for the program name and the terminal number being used. As more terminals are added to the system, more copies of TERMx are required. This would use one ID segment for each copy of TERMx plus one or more disc tracks would be needed for each swapped copy. To prevent using multiple ID segments and swap tracks, we need some way to initiate multiple input transfers to buffers in SAM. In addition we need to control these transfers by an interaction between RTE and a single program.

Using SAM for Multiple Input Buffers

Let's assume that our RTE system has three terminals that we wish to be controlled by one program. Arbitrarily assign terminal logical unit numbers of 15, 16 and 17. What we wish to do for each terminal is:

1. Output instructions to the user
   and
2. Initiate a read from the terminal.

Since the input from each terminal must be stored into a buffer in memory, we need some way to "borrow" the use of some SAM for these inputs. Further, we wish program TERM to be swapped while the input is taking place. Figure 3 illustrates the desired principles so far.

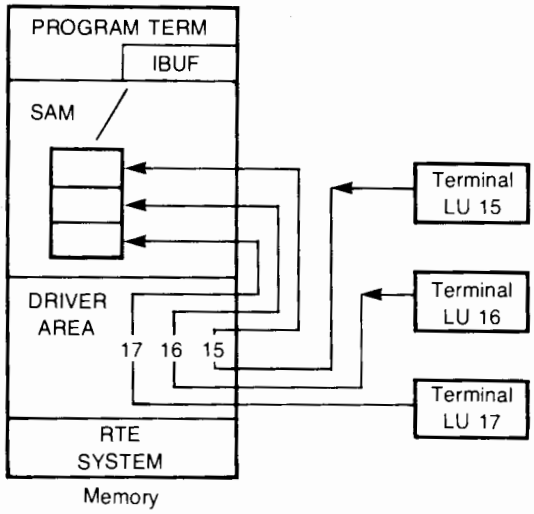


Figure 3. Using SAM For Input Buffers

Program TERM should go into an I/O suspend state until at least one of the three inputs is completed. Since SAM is used for the buffers, TERM can be swapped. Eventually all three SAM buffers will be transferred to array IBUF (one at a time). For RTE to manage SAM, some means of linking these buffers together is required. Let's link all three together in such a way that TERM can retrieve the first completed response. To facilitate the retrieval of the completed buffers a single numeric value will be associated with these linked buffers. Program TERM can then ask RTE if there are any completed buffers linked to this number. This number will be referred to as the "class number". The total class numbers available to be used are assigned during system generation. Class numbers are assigned from this table. (See Reference 2 for a detailed description of this assignment.) Programs must first request a number from RTE and then use the number throughout its execution.

RTE uses a special form of an EXEC request for allocating or using existing class numbers. Both allocation and linking are performed using this special EXEC request. An example and explanation of TERM to illustrate this follows:

```
      PROGRAM TERM
      DIEMNSION IBUF(36),ILU(3)
      DATA ILU/15,16,17/
        .
        .
        .
      ICLAS=0
      DO 400 I=1,3
      LU=ILU(I(
      WRITE(LU,200)
200   FORMAT("INPUT DESIRED TASK")
      CALL EXEC(17,LU+400B,IBUF,36,...,ICLAS)
400   CONTINUE
        .
        .
        .
```

TERM first stores the three terminal LU numbers into array ILU. During execution of the do loop, a message is written to LU 15 and the EXEC 17 sets up a class I/O read request. The EXEC 17 has three of the first four arguments the same as the EXEC 1 call. However, the EXEC 17 call has several additional arguments. ICLAS is the last of these arguments and is the only one discussed now. Since ICLAS is zero, RTE allocates a class number from the class number table which was set up at system generation. Then RTE allocates a buffer of 44 words in SAM (36 words for data plus a header of eight words to keep track of the request), performs the necessary linking and requests that the driver initiate a read on LU 15. The second time through the DO loop, ICLAS is not zero because RTE overwrote the variable address ICLAS with the allocated class number. Since the class number exists, each new buffer is allocated from SAM and linked appropriately.

The next thing TERM does is to request that a completed buffer be transferred from SAM to array IBUF. This is accomplished by TERM asking RTE to "get" the first completed buffer on TERM's class number. If no completed buffers exist, TERM has the option of either being put into a general wait state until a buffer is completed or continue other processing. If an input is complete, data is transferred to IBUF. If multiple completed buffers are ready, RTE will link them together such that TERM would get the buffers one at a time in the order they were completed (see Figure 4).

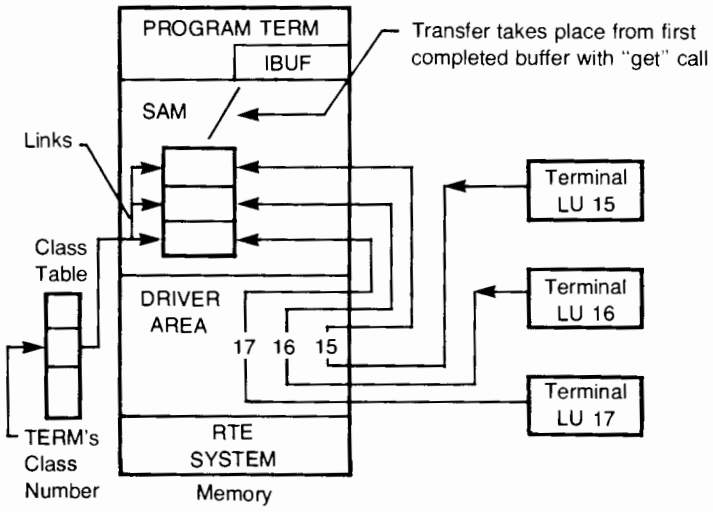


Figure 4. Using Class I/O For Input Buffering

It is important to realize that every class I/O request will require two executive requests within a program to complete a transfer: one to initiate the action and one to complete.

After program TERM gets a completed buffer to process, TERM may wish to output some acknowledgement and then initiate another input. However, which terminal should TERM use to send the response? Some way is needed to pass the originating terminal logical unit (LU) along with the data buffer. The class I/O read request (EXEC 17) has two additional arguments available for the programmer's use. These two arguments can be used for any two values the programmer desires. In the TERM example, let's use one of these arguments to pass the terminal LU along with the buffer. Perhaps we would use the second argument as a count of how many inputs have been typed from each terminal. For example:

```
      PROGRAM TERM                          terminal LU
        .
        .
      DO 400 I=1,3                            first input
      LU=ILU(I)
      WRITE (LU,200)
200   FORMAT("INPUT DESIRED TASK")
      CALL EXEC(17,LU+400B,IBUF,36,LU,1,ICLAS)
400   CONTINUE
        .
        .
        .
C  FOLLOWING STATEMENT RECEIVES DATA

      CALL EXEC(21,ICLAS,IBUF,36,JLU,KOUNT)
        .                                   input request number
        .
                                        input terminal LU
```

Note that the class I/O GET call (EXEC 21) will retrieve the terminal number in JLU and the buffer count in KOUNT. When a completed buffer is transferred on a get call, the SAM that was used is returned to RTE and the buffer is linked to the appropriate class number.

A word of caution about the EXEC 21 call. If the execution of the class I/O get call retrieves the last outstanding request, then the class number in use is returned to RTE to be reused. Subsequent get calls on the same class number will result in TERM being aborted. To prevent the release of the class number, the second parameter in the get call must have bit 14 set. This indicates to RTE that if this get call is retrieving the last request outstanding, do not return the class number.

Program TERM now examines the input request received on the get and takes whatever action is required. Another class I/O read might then be issued to that terminal for the next input. The program simply returns to the EXEC 21 call to wait for another completed buffer. The complete source code for TERM illustrates all of the principles discussed.

```
FTN4,L
        PROGRAM TERM
        DIMENSION IBUF(36),ILU(3)
        DATA ILU/15,16,17/
        ICLAS=?
        DO 400 I=1,3
        LU=ILU(I)
        WRITE(LU,200)
200     FORMAT("INPUT DESIRED TASK")
        CALL EXEC(17,LU+400B,IBUF,36,LU,1,ICLAS)
400     CONTINUE
500     CALL EXEC(21,ICLAS,IBUF,36,JLU,KOUNT)
          .
        (process request)
          .
        WRITE(JLU,600)
600     FORMAT(9NEXT?-")
        CALL EXEC(17,JLU+400B,IBUF,36,JLU,KOUNT+1,ICLAS)
        GO TO 500
        END
        END$
```

Conclusion

The intent of this article is to introduce the programmer to one method of using input transfers without wait. This is known as class I/O in RTE. While this article will not answer all the questions one might have pertaining to terminal handler programs, hopefully it will serve as food for thought in the solution of multi-terminal handlers.

This is the first in a series of articles examining the use of class I/O. Subsequent articles will cover class I/O Mailbox versus system common and class I/O double buffering.

References.

1. Anzinger, G.A. and Gadol, A.M. A Real-Time Operating System With Multi-Terminal and Batch/Spool Capabilities. Hewlett-Packard Journal, December 1975, pp. 12-20.

2. Bridges, J. RTE/II/III Class Table Structure. Computer Systems Communicator, 1976, Issue No 8, pp. 367-368.

3. Bridges, J. How to Use Class I/O and Resource Numbers in a Sort Application. Computer Systems Communicator, 1977, Issue No. 16, pp. 19-21.

INTERACTIVE DEBUGGING WITH DBUGR

Lyle Weiman

Since studies have show that debugging software often takes around half the total development time, it behooves programmers to fully utilize all the software debugging tools available to them. With the introduction of RTE-IV comes a very much more powerful program debugging tool, called DBUGR. This article will discuss the general technique of program debugging, in the context of this specific program.

There are two most-often used general debugging techniques: post-mortem and interactive. The post-mortem technique involves adding code at carefully planned locations in the program which traces the execution of that section and perhaps some of the more indicative data item values at that point. In extreme cases, it may take the form of a "crash dump": a complete printout of the program at some instant of time. Due to the large volume of printout, this latter is used most often only when the program is aborted by the operating system, and requires some support from the operating system, which RTE does not provide. In any case, a number of runs are usually required to determine the exact cause of the problem, since the information you require is like a "moving picture" of the program's execution, rather than a single "snapshot" of it.

The interactive debugging method is quite useful in a minicomputer environment partly because it requires no particular support from the operating system, and because these machines are usually so inexpensive that the overhead required to keep track of CPU and terminal connection time usage is not justified. You can therefore use a debugging package to "slow down" execution of the program so you can watch it as it runs. At a minimum, you can always examine or modify instructions, register or memory contents and control the program execution. The fancier versions allow printout in different formats (ASCII, octal, decimal, hexadecimal, as a symbolic Assembly language instruction, as a relative address, etc.), and many other extremely useful features which you'll see in this article. The important thing to note right now is that an interactive debugging package changes the speed of the program from the time frame of microseconds in which programs run, to that of seconds in which people think. This gives you time to think about what you actually told it to do (which isn't always the same as what you wanted it to do, or you wouldn't be reading this article), and perhaps change its execution as you see where it's going wrong. Let's see how you can do this with DBUGR.

First, you will need to understand Assembly language. FORTRAN programmers don't need to understand every instruction, since the FORTRAN compiler only outputs a very limited set of instructions (LDA, ADA, STA, JSB, JMP, CMA or CMA, INA or SSA or SSA, RSS or CLB or RAL and a few others in the Alter-Skip and Shift-Rotate groups), and it makes almost no use at all of the B-register or the possible combinations of Alter-Skip and Shift-Rotate groups of instructions, nor any use at all of the index registers. This subset is much smaller than the full set of which a 21MX-E computer is capable, so the task of learning these shouldn't be a problem, and the time-saving benefits will more than offset the hour or two it will take.

FORTRAN programmers will usually want to get a mixed listing (compile with the "M" option) before getting started. It is possible to get along only with the symbol table printout, and after you've developed a little knowledge of how the compiler outputs code you'll find this another time savings, because mixed listings take so long to print.

Assembly programmers already have the listing they need.

Third, you'll have to load your program with DBUGR. This is a LOADR option. DBUGR adds about 3500 words octal to the size of your program.

DBUGR may also be called directly from your program, which you may find useful. For example, you may find that a certain problem occurs only after processing a certain number of transactions. Add a line of code which calls DBUGR after that many have been processed. This will save you a great deal of time.

```
CALL DBUGR[(terminal lu)]
```

DBUGR will default to using your terminal (this value is given to it by the LOGLU subroutine).

SAMPLE DEBUGGING SEQUENCE

:RU,FTN4,&TDBUG,6,-	(see listing for program TDBUG at the end of this article)
:RU,LOADR,,%TDBUG	(load program) Note: TDBUG calls DBUGR explicitly, so the :RU,LOADR statement didn't need to force it.
:RU,TDBUG	(run program. Note: entered from MTM terminal. No LUs or other parameters passed!)
	When your program runs DBUGR will print a blank line, and the message

START DBUGR

You'll have to remember all the while you're using DBUGR that it defines its own rules for communication with it, and these are different from what you're used to with other RTE utility programs. Almost every key on your keyboard has a special meaning to DBUGR, including the carriage return and linefeed keys. It does not operate one line at a time, as do the EDITR and most others, but rather one character at a time. As each character is entered, DBUGR "sees" it and acts upon it. Some of the things you can ask it to do require only one character. As soon as it "sees" the character(s) requiring it to do something, it will do so. Immediately. Without requiring a carriage return/line-feed. DBUGR's syntax is very concise. You will find it puts very much more information on one screen than would be possible otherwise, and this is a very helpful feature. You won't need to use the scrolling features of the terminal. You will be considerably less handicapped using a terminal with little or no off-screen memory than you are using other line-at-a-time utilities.

This benefit of conciseness is achieved because DBUGR reads your input one character at a time, in binary mode so that the driver passes each character as received, without interpretation. However, the overhead of doing this is higher than line-at-a-time input. In a very active system, you may accidentally type characters faster than DBUGR can accept them, and gain system attention (either an asterisk (*) or the Multi-Terminal Monitor prompt will be typed, depending upon whether the terminal is LU 1 or not) occasionally. You should type Control-D at this point to exit (tell RTE "never mind"). You may also see that DBUGR's printout may pause momentarily at different places when used in a very busy system. You'll just have to be patient in these cases.

Unfortunately, and because of this conciseness advantage, DBUGR will not recognize the backspace character. If you make a mistake before typing the final character of a command, type DELETE (RUBOUT on some terminals). DBUGR will respond by typing an X and several spaces. The command you didn't finish and deleted will not be executed. Re-type your command.

DBUGR echoes almost all characters you input, but not all. The Escape key is echoed as a backslash (\).
DBUGR can be used with DVR00 or DVR05, but not DVR07 (multipoint).

Since the mixed-FORTRAN or Assembly listings you have contain only relative addresses (relative to where the LOADR placed each module in memory), it is very helpful to define one or more symbols to be these addresses. DBUGR allows you to enter any combination of defined symbols and constants, separated by the characters + (or blank) for additional or - for subtraction, thus saving you the trouble of doing all the octal arithmetic yourself.

For example, the symbol P is defined below to be address 30002 (the load point for the module TDBUG). In addition a breakpoint is set at relative address P+47.

```
30002<P:        P 47\B
```

Note how both commands exist on the same line. DBUGR will echo a carriage-return/linefeed to the terminal when Escape B is entered, in order to start a new line.

A "breakpoint" is the means by which you temporarily stop execution of your program in order to see how much damage has been done. When you resume execution, your program will execute at normal speed until the "breakpoint" is executed. Instructions are not interpreted by DBUGR in this mode, so you can only set a breakpoint at an executable instruction. DBUGR will temporarily save the instruction in that location when you resume execution, and store a JSB (subroutine call instruction) in that location, calling its own breakpoint-processor instruction. it is for this reason that you must never set a breakpoint:

1. at a constant, or instruction which is also used as a constant
2. at any DEF (define address)
3. at any instruction which will be configured (as in I/O) or otherwise modified
4. at any instruction which will not be executed (or you will "lose control".

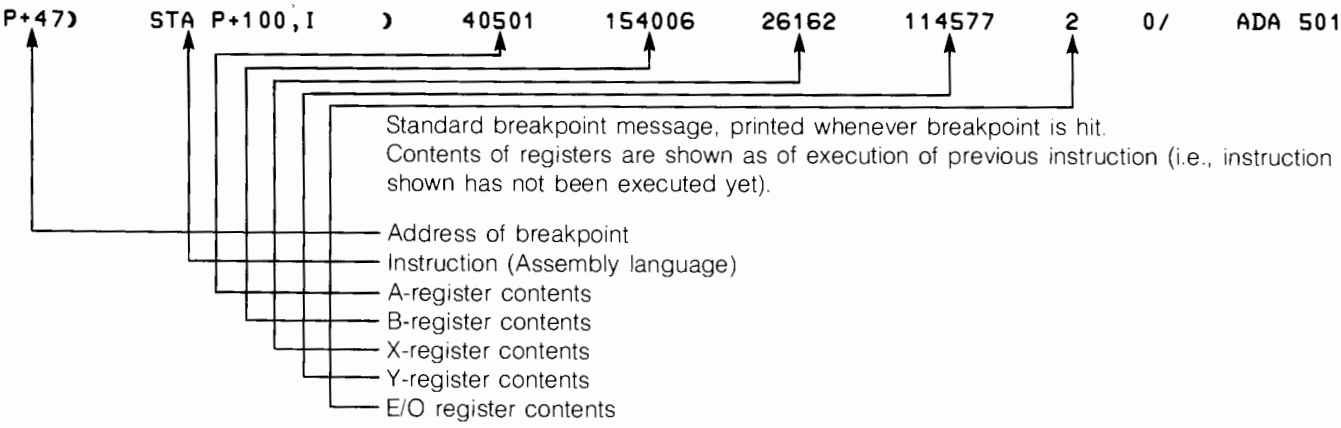
Note: you can only have one breakpoint defined at a time.

Once DBUGR has gained control, or after hitting any breakpoint, you can resume execution by typing Escape P. You must be sure your program will execute the breakpoint. Since DBUGR is not interpreting each instruction, if you've placed your breakpoint at some location that will never be executed, you have no way of recovering control of your program. The most certain, although not always the most convenient, way to do this is to set the next breakpoint at the next "decision". In FORTRAN, decisions are either IF or computed-GOTO statements. In the former, the compiler emits either an SSA (skip if the sign bit is 0) or SSA,RSS (skip if sign bit is 1). The following instruction or two will be JMP (unconditional GOTOs) which will be executed depending upon the sign bit of the A-register. DBUGR has an instruction-tracing feature which you can use to determine which way the program will go, if you'd rather not worry about keeping SSA and SSA,RSS straight. We'll show you how in a little bit.

In the computed-GOTO case, FORTRAN compiles code such as the following:

```
    JSB .GOTO
    DEF end of list
    DEF INDEX
    DEF stmt # 1
    DEF stmt # 2
     .
     .
    DEF last stnt number
end of list
```

You can set a breakpoint at the JSB, and examine the value of the index variable (INDEX in the example above). Count down that many DEFs and you'll find the DEF to the statement control will transfer to. You can move the breakpoint to that statement simply by looking at the value of the DEF. Suppose the value printed for DEF stmt # 2 was 32067 and the index variable value is 2. Then you can move the breakpoint to the next statement to be executed by typing 32067 escape B.



Suppose we want to see what's in the A-register, but there are ASCII characters in it. To examine any memory location, type its address (0 for A-register, 1 for B-register) followed by a slash (/). The contents of that location will be printed according to whatever mode is currently in effect at the time. You can control this mode, which we'll get to a little later, but you can also get a printout of the last quantity typed (either by you or by DBUGR) in other forms without affecting the print-format mode. In the example, the contents of the A-register were printed as a symbolic instruction. To find out what this is in ASCII, type a single quote (').

```
'AA''      \P
```

Note that DBUGR printed the two characters (AA) followed by a double-quote to delimit the print field, followed by three blanks. Sometimes a quantity may have only one printable ASCII character, and this helps you determine whether the character is in the high or low half-word.

On the same line above, we resume execution.

```
P+47)   STA P+100,I   )   40502   154006   27162   114577   2   0/   ADA 502
```

Since this breakpoint was set inside a DO-loop, it didn't need to be moved. This time, however, the contents of the A-register is different.

```
'AB''      \P
```

Note the increase in the contents of the A-register.

```
'AC''      3\P
```

Sometimes it's annoying to have to go through loops several times before you come to the pass you're interested in. DBUGR allows a number to be typed in front of Escape P. In the example above, the breakpoint message will only be printed after the breakpoint has been executed three times (note: the instruction which was saved when the breakpoint JSB was stored in your program will be simulated each time, as it is when you type Escape P).

```
P+47(   STA P+100,I   )   40506   154006   27162   114577   2   0/   ADA 506
```

Note the A-register has increased by three.

```
'AF''
```

INSTRUCTION TRACING

You can use instruction-tracing to "single-step" through some codes (very useful if you'd like to see where control passes in arithmetic IFS). You can't trace through REIO or EXEC calls, however. Move the breakpoint to the return address in this case.

```
\T                                  Trace one instruction.
P+50(    LDA P+61     )   40506   154006   27162   114577   2   \T
```

Standard breakpoint message is printed for instruction tracing, also. As before, instruction shown has not been executed.

Execute this instruction.

```
P+51(    ADA P+75     )   6       154006   27162   114577   2   \T
```

Note change in A-register, due to execution of previous instruction.

Execute next instruction.

```
P+52(    STA P+61     )   7       154006   27162   114577   2   \T
```

Note increase in A-register. Execute next instruction.

```
P+53(    CMA,INA      )   7       154006   27162   114577   2   \T
```

Note that contents of registers do not change with Store instructions. Execute next instruction. Next several instructions will be traced, without comment.

```
P+54(     ADA P+60      )     177771     154006     27162     114577     2     \T
P+55(     SSA,RSS     )     3     154006     27162     114577     2     \T
P+56(     JMP P+41      )     3     154006     27162     114577     2     \T
P+41(     LDA P+61      )     3     154006     27162     114577     2     5\T
```

You can trace a number of instructions. In the example, 5 instructions re traced:

```
P+42(     ADA P+76      )     7     154006     27162     114577     2
P+43(     STA P+100      )     30010     154006     27162     114577     2
P+44(     LDA P+61      )     30010     154006     27162     114577     2
P+45(     ADA P+77      )     7     154006     27162     114577     2
P+46(     ADA P+101      )     40510     154006     27162     114577     2
```

EXAMINING MEMORY LOCATIONS

```
P 17/     LDA P+60
```

Examine relative location P+17 (octal). Current printout mode is symbolic, so value is printed as instruction.

You can examine next location by typing Linefeed key (or Control and J keys held down together). Several locations are shown. Note that no character is echoed, but each display starts on new line.

```
P+20/     STA P+61
P+21/     LDA P+63
P+22/     STA P+62
P+23/     LDA P+65
P+24/     STA P+64
P+25/     DLD
```

The next location will be an address. Addresses can be displayed with the underline key (—).

```
P+26/     IOR 72     -P+70
P+27/     DST
P+30/     IOR 70     -P+66
P+31/     JSB 224,I    224/    ISZ 1523    =35523    -P+5521
```

Here, the effective address is desired. First, 224 is examined. Its contents are then printed as an address.

Note what happens when you type linefeed (Control-J) now:

```
225/    ISZ 1661
```

You get the location following 224, not P+32. You'll have to type P 32 explicitly in order to resume examination of that section of memory.

```
P+33/    IOR 76    -P+74
P+34/    JSB 223,I    =114223
P+35/    DST
P+36/    IOR 74    -P+72
```

CHANGING PRINT MODE

You can change the print mode so that data you examine will be printed as constants by typing Escape C.

```
\C    P 17/    62062
 P+20/    72063
 P+21/    62065
 P+22/    72064
 P+23/    62067
 P+24/    72066
 P+25/    104200
 P+26/    30072
 P+27/    104400
 P+30/    30070
```

CHANGING NUMBER BASES

You can change the radix which DBUGR uses to print numbers. With radix 8 you get octal, with radix 10 numbers are printed in decimal, and with radix 16 they're printed in hexadecimal. DBUGR will accept numbers from you in only two radixes, however: 8 and 10. Decimal numbers must be suffixed by a period. Remember this when changing radixes.

DBUGR will accept a radix anywhere from 2 to 31, althrough 2, 8, 10 and 16 are the only really useful ones.

```
P+31/   114224   16.\R
P 17/   6432
P+10/   7433
P+11/   6435
P+12/   7434
P+13/   6437
P+14/   7436
P+15/   8880
P+16/   303A
P+17/   8900
P+18/   3038
P+19/   9894
```

Note how the address is also printed in hex:

```
P+1A/   3038

P+1B/   303E
P+1C/   9893
P+1D/   8900   10.\R
```

Now, change to decimal

```
P 17/   25650.
P+16./   29747.
P+17./   25653.
P+18./   29748.
P+19./   25655.
P+20./   29750.
P+21./   34944.
P+22./   12346.
P+23./   35072.
P+24./   12344.
P+25./   39060.
P+26./   12344.
P+27./   12350.
P+28./   39059.
P+29./   35072.
P+30./   12348.
P+31./   25663.
P+32./   29747.   3.1\R
```

Just for laughs, let's try base-31.

```
P 17/   QLD
P+G/   UTI
P+H/   QLG
P+I/   UTJ
P+J/   QLI
P+K/   UTL
P+L/   1587
P+M/   CQ8
P+N/   15FB
P+O/   CQ6
P+P/   19K0
P+Q/   CQ6
P+R/   CQC
P+S/   19JU
P+T/   15FB
P+U/   CQA
P+10/   QLQ
P+11/   UTI
P+12/   QLE
```

DBUGR has several print modes, and their relative permanence can be manipulated. The least permanent is the temporary, which lasts only for the execution of the current command. For example, the apostrophe prints the previous quantity typed in ACSII. Another temporary print mode commands are the exclamation mark, which prints the last quantity types in symbolic form:

```
P+1A/    15I    !ADA P+1U
P+1B/   UTI    !STA P+1I
P+1C/    1IL    !CMA,INA
```

The "last quantity typed" refers to the last value, symbol, or expression typed, either by DBUGR or by you.

You already saw how the underline can be used to print a value as an address.
The equality (=) key prints the last quantity in whatever radix is in effect.

```
P+1D/    I55    !ADA P+1H    =I55    8.\R
```

Return to base-8


```
P+55/    2021
P+56/    26043
P+57/    26021    ',''
```

OCTAL/DECIMAL CONVERSIONS & CALCULATIONS

You can use DBUGR to do your octal or decimal arithmetic for you, or to convert from octal to decimal or hex, and from decimal to octal. You can convert from hex to decimal, if you're willing to go through the manual procedure of converting to octal first.

For example, determine the absolute address of location 66 relative to a module loaded at location 32477. A subtraction is shown next just as an example. Next, 55 (decimal) is converted to octal

```
32477 66-32565    32565-66-32477   55.-67
```

Here, a decimal number is added to an octal number. The result is printed in octal.

```
128. 67+267
```

(To get the result in decimal, change the radix).

TO RE-EXECUTE A SECTION OF CODE, OR SKIP A SECTION

Use the address you want to start at, followed by Escape G. Be sure your breakpoint will be executed.

For example:

```
P 47\B                    Set breakpoint

   \P                     Proceed

P+47(   STA P+100,I   )   40501   154006   27162   114577   2   \P
P+47(   STA P+100,I   )   40502   154006   27162   114577   2   \P
P+47(   STA P+100,I   )   40503   154006   27162   114577   2   P 17\G
```

Go back to relative P 17

```
P+47(   STA P+100,I   )   40501   154006   27162   114577   2   \P
```

Note that DO-loop has been restarted.

```
P+47(   STA P+100,I   )   40502   154006   27162   114577   2   \P
P+47(   STA P+100,I   )   40503   154006   27162   114577   2
```

PATCHING MEMORY

Typing any constant, symbolic instruction, address or ASCII characters after examining any location, followed by linefeed (Control-J on 264x terminals) or carriage return will store that data item in the examined location (subject to the limitations imposed by memory-protect and DMS). Note that the Control-J is indicated only by the start of a new line.

```
P 65/    AA 502    'AB''  AC''
```
Examine location 65 (relative)

Print as ASCII.
Store ASCII "AC" in this location (linefeed or Control-J used after quote is not echoed. Next location is printed on next line).

Note: you can go back and examine the previous location by using the "up-arrow" key (↑).

```
P+66/    0    1
```
Change relative location 66 from 0 to 1

```
P+67/    0    2
```
Change relative location 67 from 0 to 2

```
P+70/    ADB 631
```

Advance to location 71 without modifying location 70 (no new expression is typed, only linefeed is entered).

```
P+71/    JSB 1004,I
```
Advance to next location

```
P+72/    0    10.
```
Change relative 72 to decimal 10.

```
P+72/    0
```

DBUGR will also accept any symbolic instruction (memory-reference, ASG, SRG, etc.). In the case of memory-reference instructions which cross a page boundary, you must provide your own link. The easiest way to do this is to look for another memory-reference instruction referencing the same location you want, and see which link it uses. Look for references in the direction farthest from the location itself, in order to find a reference which already has a link (these references are more likely to be on a different memory page than those closer to the referenced item).

REMOVING THE BREAKPOINT

```
\B
```
Entering Escape then B removes the current breakpoint.

```
\P
```
Proceeding without any breakpoint causes the printout:

```
END DBUGR
```
This message indicates that DBUGR has been exited, and will not be re-entered unless it's called explicitly, or the program is re-scheduled after terminating without saving resources. However, the program continues execution:

```
07>ST,TDBUG
```
Note that the program is still running:

```
  99 1 0      0   0   0   0   0
```

EXITING THE HARD WAY

If you wish to abandon the debugging effort, and exit the program in the simplest way possible, simply force a memory-protect violation:

```
3\G
 ABEND  TDBUG ABORTED
:
```
User's FMGR copy sends its prompt character.

Only a few of the most useful features of DBUGR have been shown. These and many others are documented in the DBUGR Manual, part number 92067-90005. You are encouraged to try the above examples first, and then those in the manual.

> Note: If you record your patches (display all symbolic instructions you change in octal), you can use a routine in the Contributed Library called PMOD, which will permanently place your patches in your program. On subsequent executions of the program, they will be there.

SUBTLE DISTORTIONS INTRODUCED BY THE DE-BUGGING PROCESS

Interactive debugging may also introduce a subtle form of distortion to your program if the bug you're looking for is at all related to some other event or events.

As a very simple example, suppose the bug you're after is related to the simultaneous access of data by two programs:

```
PROGRAM A(3,20)
COMMON I,J,K
 .
 .
 .
 .
IF(I .EQ. 3) I = I + 2
 .
 .
 .
 .
PROGRAM B(3,90)
COMMON I,J,K
 .
 .
 .
 .
IF(I .EQ. 3) I = I + 3
 .
 .
 .
 .
```

Suppose, in this example, that programs 'A' and 'B' share system common, so that I, J and K in each are the same. Suppose both are resident in different partitions. Suppose that the initial value of I is 3, and that 'A' runs but is interrupted after it has tested I, but before it has begun the **I = I + 3** calculation, and swapped out by a higher-priority program. The swap will suspend execution of 'A', but since 'B' was assumed to be resident in a different partition, it can run (let's assume it does). 'B' also tests the value of I, finds the condition true, changes I's value to 6, and continues. When 'A' resumes execution again, it will add 2 to the present value of I, leaving it 8. Note how the final value of I is extremely dependent upon how programs 'A' and 'B' run with respect to each other. If 'A' ran separately, the final result would be 5. If 'B' ran separately, the final result would be 6. Can you figure out any other possible values?

Use of an interactive debugging package on either 'A' or 'B' would likely slow one of them down so much that the only values of I the programmer would ever see would be 5 or 6. The programmer would then be climbing the walls because he knows he's got an intermittent problem, and he can't find it when he looks for it.

Besides shared resources (which the knowledgeable programmer will control via resource numbers), there may be other cases where your program may execute slightly differently depending upon events external to it, and possibly to the computer. Examples of events which are internal to the machine and can influence program execution are:

1. Contention for I/O devices. Even if your program has priority 1, it may have to wait for a device to complete a previous request.

2. Swapping of other programs (even lower/priority programs, because the CPU is slowed down during any DCPC transfer).

3. Contention for system available memory (SAM). SAM is used for re-entrant subroutines, so that the mere fact that a program calls a routine which happens to be in the re-entrant library can sometimes suspend it for unavailable memory.

4. The temporary unavailability of tracks on LU 2 or 3 may prevent a lower-priority program from being swapped out in order to make room for a higher-priority one. The lower-priority program will continue to run until the condition clears itself or the program terminates. Note that a termination call which specifies that resources be saved will tie up that partition indefinitely, in this case.

In addition to the above, loading the program with DBUGR may change the points at which your program crosses page boundaries, which will add indirect references to some instructions, and remove them from others. Since indirects take an extra memory cycle, your program will run slightly slower in some places, and slightly faster in others, merely because DBUGR has been added. However, these effects are minute compared to the others above, and in all likelihood would never be noticeable.

Any of the items above may introduce delays into your program which will very likely be different and occur at different points in the program's execution with each run.

How does one proceed in the event the problem can't be found through the normal debugging techniques? One proceeds mainly by instinct, limiting DBUGR commands as much as possible, in order to minimize delays. If at all possible, determine at some gross level that something always happens in a certain way that can be tested. For example, when the condition occurs, a location may always be set to some value it normally is not supposed to have, or some value it may have, but infrequently. Add code to the modules which utilize this location to test for this value, and call DBUGR when they've found it.

Another technique, called transaction logging, may be helpful when those shown above fail. It involves modifying your programs so that, as each new "transaction" arrives for processing, some trace of it is left somewhere. SAM is a convenient place to do this, but every effort must be made to minimize the amount of SAM which is tied up in this manner, because this may introduce an additional delay if SAM is a scarce resource. Transaction-traces written into SAM should be removed very quickly by a high-priority program which locks itself into a partition, reads these buffers and writes them to the disc. Data blocking should be used to minimize the disc accesses. However, beware of making this program too large, or the partition it uses will be large, and hence it will effect the competition among large programs for partitions, and introduce a different form of distortion.

There should be a program which can analyze these transactions later, after the problem has already evidenced itself, and therefore its own effect on the system can be ignored. That program may be quite fancy, but need only print out all transactions on a printer. If you suspect that there may be hundreds or thousands of transactions going through the system during the sample period (as would be the case in problems which take days or weeks to show themselves), it would be far better instead to format the data into an ASCII file with transaction number and time of day in it, so that the EDITR can be used to select only relevent protions for printout. Circular files are useful because data which is too old is automatically written over, but you are limited to fixed-size records.

In later articles, we will discuss the techniques sometimes employed to find intermittent bugs. Most of them are extremely dependent upon the characteristics of the programs themselves and how the bug manifests itself, and depend to a great degree upon a thorough understanding of the software itself and everything it depends on, including, in some cases, the hardware. They will therefore often be different each time, but it is hoped that you'll be able to adapt one of them to the job at hand as required.

Happy Hunting!

```
PAGE 0001  FTN.  12:20 PM  THU.,  5  JAN., 1978

 0001  FTN4,L,M
 0002        PROGRAM TDBUG
 0003        INTEGER J1(10)
 0004        CALL DBUGR
 0005  5     CONTINUE
 0006        I=10
 0007        J=3
 0008        JJ=2HAB
 0009        X=2.3
 0010        XX=X**2
 0011        DO 10 I=1,10
 0012        J1(I) =2HAA+I-1
 0013  10    CONTINUE
 0014        GOTO 5
 0015        END

 FTN4 COMPILER: HP92060-16092 REV. 1805

 **  NO WARNINGS **  NO ERROPS **    PROGRAM = 00066      COMMON = 00000

PAGE 0002  TDBUG 12:20 PM  THU.,  5  JAN., 1978

 0002        PROGRAM TDBUG
 0003        INTEGER J1(10)
 0004        CALL DBUGR
                       J1      BSS 00012B
        00012  000000          NOP
        00013  000001X         JSB CLRIO
        00014  000015R         DEF *-2+00003B
 0005  5     CONTINUE
        00015  000002X         JSB DBUGR
        00016  000017R         DEF *-4+00005B
 0006       I=10
 0007       J=3
        00017  000060R @5       LDA 00060B
        00020  000061R          STA I
 0008       JJ=2HAB
        00021  000063R          LDA 00063B
        00022  000062R          STA J
 0009       X=2.3
        00023  000065R          LDA 00065B
        00024  000064R          STA JJ
 0010       XX=X**2
        00025  000003X          JSB .DLD
        00026  000070R          DEF 00070B
        00027  000004X          JSB .DST
        00030  000066R          DEF X
 0011       DO 10 I=1,10
        00031  000005X          JSB .RTOI
        00032  000066R          DEF X
        00033  000074R          DEF 00074B
        00034  000006X          JSB ERR0
        00035  000004X          JSB .DST
        00036  000072R          DEF XX
        00037  000075R          LDA 00075B
        00040  000061R          STA I
```

```
0012          J1(I) =2HAA+I-1
        00041  000061R          LDA I
        00042  000076R          ADA 00076B
        00043  000100R          STA A.001
0013  10     CONTINUE
        00044  000061R          LDA I
        00045  000077R          ADA 00077B
        00046  000101R          ADA 00101B
        00047  100100R          STA A.001,I
0014          GOTO 5
        00050  000061R @10      LDA I
        00051  000075R          ADA 00075B
        00052  000061R          STA I
        00053  003004           CMA,INA
        00054  000060R          ADA 00060B
        00055  002021           SSA,RSS
        00056  000041R          JMP 00041B
0015          END
        00057  000017R          JMP @5
        00060  000012           OCT 000012
                         I      BSS 00002B

PAGE 0003  TDBUG 12:20 PM  THU.,  5  JAN., 1978

        00063  000003           OCT 000003
                         JJ     BSS 00001B
        00065  040502           OCT 040502
                         X      BSS 00002B
        00070  044631           OCT 044631
        00071  115004           OCT 115004
                         XX     BSS 00002B
        00074  000002           OCT 000002
        00075  000001           OCT 000001
        00076  177777R          DEF 77777B
        00077  040501           OCT 040501
                         A.001  BSS 00001B
        00101  177777           OCT 177777

PAGE 0004  TDBUG 12:20 PM  THU.,  5  JAN., 1978

 SYMBOL TABLE

 NAME       ADDRESS     USAGE                TYPE       LOCATION

 @10          00050R    STATEMENT NUMB
 @5           00017R    STATEMENT NUMB
 CLRIO        00001X    STATEMENT FUNCTION   REAL       EXTERNAL
 DBUGR        00002X    STATEMENT FUNCTION   REAL       EXTERNAL
 ERRO         00006X    STATEMENT FUNCTION   REAL       EXTERNAL
 I            00061R    VARIABLE             INTEGER    LOCAL
 J            00062R    VARIABLE             INTEGER    LOCAL
 J1           00000R    ARRAY(*)             INTEGER    LOCAL
 JJ           00064R    VARIABLE             INTEGER    LOCAL
 X            00066R    VARIABLE             REAL       LOCAL
 XX           00072R    VARIABLE             REAL       LOCAL

 PAGE 0005  FTN.  12:20 PM  THU.,  5  JAN., 1978

  0016         END$
\END
```

CAUTION ON MATH OPERATIONS ON HOLLERITH CONSTANTS

Jim Bridges

Many FORTRAN IV programmers discover short cuts which are not specified in the ANSI standards for that language. In some cases, the short cuts will violate the standard and yet will work because of the different implementation on various computers. This article covers one short cut which was discovered not to work. In this case, the programmer was apparently not familiar with the standard, found a situation which did not work and reported it as a bug. Very few people who program in FORTRAN IV have actually read the ANSI standards (they are considerably less exciting than a novel). In most cases, this does them no harm and there probably are some excellent programmers who never heard of ANSI standards. Regardless, this article may show how to identify some problems with your code.

Look at the following sample program, which uses DATA statements to fill in DOUBLE PRECISION variables with HOLLERITH information.

```
FTN4,M
      PROGRAM EQDMT(3,89),EXPERIMENT WITH HOLLERITH IN DO NUMBER
      DIMENSION NAME1(3),NAME2(3),LU(5)
      DOUBLE PRECISION FN1,FN2
      EQUIVALENCE (FN1,NAME1),(FN2,NAME2)
      DATA FN1/6H!S4L07/,FN2/6H!S4L67/
      CALL RMPAR (LU)
      IF (LU.EQ.0) LU = 1
      WRITE (LU,1000) ((NAME1(I(,I=1,3),(NAME2(I),I=1,3),J=1,2),FN1,FN2
10000 FORMAT (2(3X,3A2/),/2(3X,306/),/2(3X,G20.15/))
      IF (FN1.EQ.FN2) WRITE (LU,1010) (NAME1(I),I=1,3),(NAME2(I),I=1,3)
      IF (FN1.LT.FN2) WRITE (LU,1020) (NAME1(I),I=1,3),(NAME2(I),I=1,3)
      IF (FN1.GT.FN2) WRITE (LU,1030) (NAME1(I),I=1,3),(NAME2(I),I=1,3)
1010  FORMAT (3A2," IS EQ TO "3A2)
1020  FORMAT (3A2," IS LT TO "3A2)
1030  FORMAT (3A2," IS GT TO "3A2)
      END
      END$
```

While it is perfectly legitimate to use HOLLERITH data to initialize DOUBLE PRECISION variables, according to ANSI standard the variable becomes undefined. This is reasonable because the variable is not a string variable, even though it may sometimes be used as a string. (In fact, one of the criticisms of FORTRAN IV is the lack of string handling features.) Therefore the operations of .EQ.,.LT., and .GT. are arithmetic operations — not string operations. As such, double precision arithmetic is used to make the comparisons requested.

As noted in Appendix A of the HP FORTRAN IV manual, a double precision number is significant to 11 or 12 decimal digits, depending upon the magnitude of the leading digit in the fraction. The memory layout of double precision format is:

```
S!14              0    WORD 1 (S=SIGN OF FRACTION)
15                0    WORD 2
15      8!7    1!F     WORD 3 (F=SIGN OF EXPONENT
                       BITS 1 - 7 IS EXPONENT
```

A mixed listing of the above program will show that the numbers implied by the above HOLLERITH information differ by approximately 10 exp −34. Thus, within the accuracy of the double precision arithmetic, the two values are equal. Therefore the statement with format 1010 will cause a printout. Note that, because the sign of the exponent is in the least significant bit, that a change in this bit will cause a large variation in value.

Because of the much simpler format of the integer, problems of this sort do not occur when filling integer variables with HOLLERITH information.

BIT BUCKET

Samantha wishes to correct an error in the last issue, which described the method of using the symbols **.ZRNT** and **.ZPRV** when writing subroutines which might go into the memory resident library. The statement was made that a subroutine which was placed in the memory resident library would have a copy also placed in the disc resident library. The subroutine will also be placed in the disc resident library only for RTE IV (not RTE II or RTE III).

In a RTE II/III system, a disc resident program which calls a memory resident subroutine will cause a memory protect violation, which is examined by the system for legality. If the violation is JSB (call) to an address within the bounds of the memory resident library, it is permitted. However, this process adds overhead to the system: depending upon the particular CPU, it can cost up to 1 millisecond to "break" the memory protect fence.

Because of this overhead, it is best to consider carefully what programs are made memory resident. All the type 6 subroutines referenced by the memory resident programs will get placed in the memory resident library and these may also be subroutines commonly used by disc resident programs.

Samantha invites you to write in requesting information on any technical question regarding HP 1000 system software. You will always receive a personal replay, whether or not your question is answered in this column. Address your questions to:

SOFTWARE SAMANTHA
c/o HP 1000 Communicator EDITOR
Hewlett-Packard Data Systems
11000 Wolfe Road
Cupertino, California 95014

DETECTING PROBLEMS AT BOOT-UP TIME

Jim Bridges

A knowledge of what happens at boot-up time gives the user a powerful tool to help trouble-shoot systems when a hardware failure is suspected. The kind of failure this applies to is one which may occur at boot-up itself. Bootstraps are, by necessity, very simple programs and they do minimal checks. Hence it is possible to have a bootstrap seem to work, that is, it seems to bring in the system and the system "runs" but is doesn't run very well. It may give unusual memory protects or dynamic mapping errors, for example, which make no sense. In such a case, perhaps the system was not completely brought into memory or perhaps a parity error occurred during the boot process.

A description of the boot process is contained in the May 1976 issue of the Communicator in the article "Know Your RTE, Part 2". New information is added here as well as (it is hoped) a simpler presentation.

There are two parts to the bootstrap program:

1. A bootstrap which is "system independent" is placed in the upper 64 words (but never higher than 32K) of memory. The bootstrap may be loaded from ROM, toggled in from the front panel or loaded from some other medium. For simplicity, we will call it the "ROM Boot".

   The ROM Boot loads, using a specified head number, the information beginning at sector zero, cylinder zero. It does not know whether there is a system on the disc or not. It begins loading into memory starting at location 2011. It issues a request to read a word count greater than a track (not possible in a single request) and then jumps to the contents of memory address 2055 as soon as the disc has processed the command. That is, DMA transfer is not checked. It is assumed that only the first few sectors are of interest and they will have been loaded into memory whether or not DMA completes.

2. The information that the ROM boot brings off cylinder zero can be called the "RTE boot". Although it bears no special relationship to the system itself, it contains a table which it is used to identify the portion of the disc to be loaded into memory and where it will go in memory. For RTE II and RTE III the map looks like this:

   ```
   000002  002000    0TTTSS  (for Base Page)
   002000  MMMMMM    0TTTSS  (for System)
   ```

   where

   MMMMMM = Last word address plus one of system
   TTT = Track where base page or system starts on disc
   SS = Starting sector on TTT

   For a typical system which begins on cylinder 0, the starting sector is 2 for base page and 22 (octal) for the system. The track is zero in this case. Note that the RTE bootstrap itself is on sectors 0 and 1 and that all sectors have 64 (decimal) words.

A listing of the RTE bootstrap for the 7900 disc is given at the end of this article. With this listing, it is relatively simple to understand the "slow-boot" process described in the previously referenced Communicator article. This listing is not available else in customer documentation. However, the listing of the ROM boot is given in the Loader ROM's Installation Manual, 12992-90001.

The slow-boot process can be described as:

1. Load the ROM boot into memory.
2. Patch the ROM boot to cause a halt before it executes the RTE boot which it brings in from sectors 0 and 1. Location is patch is 077775 (or lower if less than 32K). Then execute the ROM boot from its starting location (usually 77700). When the halt occurs, the RTE boot is in memory (if the hardware is working).
3. Patch the RTE boot so that it halts before jumping into the system code. Location to patch is 2016 (true for 7900/7905/7920 discs). Execute the RTE boot from 2175 (see listing below).

Notice that the first action by the RTE boot is a halt 77 at location 2200. This is patched to a NOP (zero) by the SWTCH program. It may be patched back to 102077 if you wish, in which case it would not be necessary to patch 077775 to a halt for a slow-boot.

The next action taken by the RTE boot is to move itself to high memory starting at 77750. Then it jumps to the start of the relocated boot (i.e., 77750). This is done because the system itself would otherwise overlay the RTE boot. Because of memory wrap-around, if the computer has less than 32K of memory the RTE boot will still be moved near the top of memory.

Because the RTE boot is moved, the halt inserted at 2016 will actually execute in high memory. When it occurs, you may make patches to the system, if you wish. To check if the system was loaded correctly into memory, examine a few locations in the communications area of the base page (1647 to 1777). Usually, it is necessary to verify only one or two locations. Easy ones to check are 1651 (number of EQT entries), 1653 (number of logical units), 1674 (I/O select code of the time base generator).

This minimal check will detect a great many problems, but if it does not show anything wrong, then you may wish to consider the effect of a parity error at boot time.

There is a memory parity error INT/IGNORE HALT switch on the computer (see installation and service manual for location and instructions) which determines what happens when a parity error occurs. If the switch is in the HALT position, a memory parity error causes the computer to halt at location 5. In the INT/IGNORE position, the effect of a parity error is to create an interrupt which (like all interrupts) is processed by the RTE system. If the switch is in the INT/IGNORE position, you will not be able to "see" the occurrence of parity errors until the system is actually running: hence parity errors at boot-time are "lost". If such errors go undetected at boot-time, they may cause very strange symptoms, depending upon what part of memory is bad.

The processing of a parity error in RTE II and RTE III is very minimal. When one occurs, the system halts with 102005 in the T register and the memory violation address in the B register. In RTE III the violation address may be in one of four maps, but it is most likely to be in the user map. If the switch is in the HALT position, then the operator must toggle in an instruction (e.g., LIA 5) to pick up the violation address. This is annoying or difficult enough for most people that they choose the INT/IGNORE position of the switch. On the other hand, the processing by the system is so minimal that many people feel its better to choose the HALT position. Whatever your choice, when memory parity problems seem frequent, it is best to stick with the HALT position for a time.

The listing for the 7900 RTE boot is given below.

```
Memory Instr        Mnemonics                 Comments

02011 067652  START LDB MADR          GET MAP ADDR
02012 077641        STB T3            SAVE
02013 017506        JSB BGN           LOAD FROM DISC
02014 017506        JSB BGN
02015 017506        JSB BGN
02016 124003        JMP 3,I           GO INITIALIZE SYSTEM IN MEMORY
*
02017 002011  BGN   OCT 2011          (USED AS ADDR BY CODE AT L1)
02020 167641        LDB T3,I          GET FWA IN MEM
02021 037641        ISZ T3            BUMP PTR
02022 163641        LDA T3,I          GET LWA IN MEM
02023 037641        ISZ T3            BUMP TO DISC ADDR
02024 003304        CMA,CCE,INA       2'S COMPL OF LWA
02025 040001        ADA 1             + FWA GIVES NEG WORD COUNT
02026 005225        RBL,ERB
02027 106702        CLC 2             PREPARE FOR DMA CONTROL WORD #2
02030 106602        OTB 2             OUTPUT FWA OF MEMORY FOR LOAD
02031 073640        STA T2            SAVE 2'S COMPLEMENT OF WORD COUNT
02032 163641        LDA T3,I          GET DISC ADDR
02033 013636        AND M177          MASK FOR SECTOR
02034 070001        STA 1             SAVE IN B REGISTER
02035 123641        XOR T3,I          BLANK OUT SECTOR TO GET TRACK
02036 037641        ISZ T3            BUMP TO NEXT POSITION IN MAP
02037 001727        ALF,ALF           PUT TRACK IN BOTTOM BYTE
```


```
Memory Instr        Mnemonics          Comments
02040 001200        RAL
02041 043643        ADA STRK#          AND ADD STARTING TRACK FOR SYSTEM
02042 073633        STA CT1            SAVE STARTING CYL FOR SEEK COMMANDS
02043 005100        BRS                DIV BY 2: OFFSET #BLKS FROM STRT 1ST TR
02044 060001        LDA 1
02045 001727        ALF,ALF            MUL BY 128 TO COMPUTE WDS TO SKIP
02046 001300        RAR
02047 003004        DMA,INA            2'S COMPLEMENT
02050 043642   R1   ADA WDTRK          ADD NUMB WDS TRACK
02051 073636        STA T6
02052 003004        CMA,INA            2'S COMPLEMENT
02053 073637        STA T7
02054 002001        RSS                ALWAYS SKIP (NEXT LOC IS AN ADDRESS)
02055 002175        OCT 2175           THIS IS ADDRESS TO GO WHEN STARTING BOOT
02056 063640        LDA T2             GET WD CNT REMAINING ON TRACK
02057 002021        SSA,RSS            DONE IF NO MORE WDS TO LOAD THIS PART
02060 127506        JMP BGN,I
*
02061 043636        ADA T6             ADD # WDS TO LOAD THIS TIME
02062 073640        STA T2             SAVE REMAINDER FOR NEXT TIME THRU LOOP
02063 002020        SSA                IS THIS LAST LOAD?
02064 002400        CAL                YES
02065-043637        ADA T7             A = 2'S COMPL OF WD COUNT THIS LOAD
02066 102702        STC 2              PREPARE DMA CHANNEL
02067 102602        OTA 2              OUTPUT WORD COUNT
02070 063633        LDA CT1            GET CYL ADDRESS FOR SEEK
02071 102612        OTA 12B
02072 103712        STC 12B,C          SEND TO DATA CHANNEL
02073 063644        LDA SEEKC          GET SEEK COMMAND
02074 106713        CLC 13B            CLEAR COMMAND CHANNEL
02075 102613        OTA 13B
02076 103713        STC 13B,C          OUTPUT SEEK COMMAND TO COMMAND CHANNEL
02077 047646        ADB M24            THESE NEXT INSTRUCTIONS
02100-006021        SSB,RSS            BUILD THE HEAD SECTOR
02101 047634        ADB BIT8           ADDRESS
02102 006020        SSB                FOR THE SEEK
02103 047645        ADB D24            COMMAND
02104 047647        ADB OFSET          = HEAD #2 IF ON LOWER SUBCHNL
02105 102312        SFS 12B            FIRST ADDR WORD OF SEEK ACCEPTED?
02106 027574        JMP *-1
*
02107 106612        OTB 12B            OUTPUT HEAD SECTOR ADDRESS
02110 103712        STC 12B,C
02111 063650        LDA READC          GET READ COMMAND
02112 102313        SFS 13B            SEEK COMPLETE?
02113 027601        JMP *-1
*
02114 102613        OTA 13B            OUTPUT READ COMMAND
02115 103712        STC 12B,C          TO DATA CHANNEL
02116 106713        CLC 13B            CLEAR COMMAND CHANNEL
02117 103706        STC 6,C            ACTIVATE DMA
02120 103713        STC 13B,C          INITIATE ACTUAL READ OPERATION
02121 102313        SFS 13B            READ COMPLETE?
02122 027610        JMP *-1
*
02123 102106        STF 6              ABORT ANY DMA IN PROGRESS
02124 103712        STC 12B,C
02125 063651        LDA STWD           GET STATUS CHECK COMMAND WORD (ZERO)
```

```
 Memory Instr          Mnemonics                 Comments
 02126 106713          CLC 13B
 02127 102613          OTA 13B
 02130 103713          STC 13B,C       INITIATE STATUS CHECK COMMAND
 02131 102312          SFS 12B         STATUS READY?
 02132 027620          JMP *-1
*
 02133 102512          LIA 12B         GET STATUS
 02134 000010          SLA             ANY ERRORS?
 02135 102031          HLT 31B         YES! HALT. PRESS RUN TO RESTART
 02136 000010          SLA
 02137 027500          JMP START
*
 02140 006400          CLB
 02141 037633          ISZ CT1         BUMP CYL ADDR FOR NEXT SEEK
 02142 002400          CAL
 02143 027537          JMP R1          AND GO LOAD NEXT TRACK
*
 02144 177600  CT1     DEC -128        COUNTER TO MOVE 128 WORDS
 02145 000400  BIT_    OCT 400
 02146 000177  M177    OCT 177
 02147 000000  T6      NOP
 02150 000000  T7      NOP
 02151 077500  T2      OCT 77500       START OF RELOCATED BOOT
 02152 077500  T3      OCT 77500       FWA OF RELOCATED BOOT
 02153 014000  WDTRK   DEC 6144        # WORDS TRACK
 02154 000000  STRK#   NOP             ACTUAL STARTING TR OF SYS (MAY BE#0)
 02155 030000  SEEKC   OCT 30000       SEEK COMMAND WORD
 02156 000030  D24     DEC 24          # SECTORS/SURFACE
 02157 177750  M24     DEC -24         # OF 128 WD SECTRS/SURFACE
 02160 001000  OFSET   OCT 1000        HEAD #2 OR 0 IF ON SUBCHNL 1
 02161 020000  READC   OCT 2000        READ COMMAND
 02162 000000  STWD    OCT 0           STATUS COMMAND WORD
 02163 077653  MADR    OCT 77653       FWA OF MEMORY/DISC MAP TO LOAD SYSTEM
*
 02164 000002          OCT 2           FWA ON BASE PAGE
 02165 002000          OCT 2000        LWA + 1 ON BASE PAGE
 02166 000002          OCT 2           DISC TR SECTOR WHERE BP LOCATED
 02167 002000          OCT 2000        FWA OF MAIN MEMORY
 02170 056032          OCT 56032       LWA + 1 OF MAIN MEMORY
 02171 000022          OCT 22          DISC TR SECTOR WHERE SYSTEM LOCATED
*
 02172 000000          NOP             NOT USED
 02173 000000          NOP             NOT USED
 02174 000000          NOP             NOT USED
 02175 000000  GO      NOP             THIS IS WHERE WE BEGIN EXECUTION OF THE
 02176 102106          STF 6           ABORT DMA
 02177 107700          CLC 0,C         INTERRUPTS OFF
 02200 102077          HLT 77B         (PATCHED BY SWITCH TO NOP)
 02201 162017  L1      LDA BGN,I       GET CONTENTS OF BOOT
 02202 172151          STA T2,I        MOVE TO MEM STARTING AT 77500
 02203 036017          ISZ BGN         BUMP SOURCE PTR
 02204 036151          ISZ T2          BUMP DEST PTR
 02205 036144          ISZ CT1         BUMP CNTR (SET AT 128 WORDS)
 02206 026201          JMP L1          CONTINUE LOOP
*
 02207 026152          JMP T3 ,I       GO TO START OF RELOCATED BOOT (77500)
                       END START
```

PATCH A SYSTEM BEFORE YOU INSTALL IT

Jim Bridges

System generation often takes several hours. Therefore, it is desirable not to have to repeat the process if only minor errors have occurred. For example, you may have specified the wrong select code for a driver. In such cases, it is preferable to patch the system rather than regenerate.

If you have used the on-line generator for RTE II or RTE III, then the system exists in a type 1 file. By patching the file, you effectively patch the system before installation rather than at boot-up or after the system is running. This procedure can be a great deal easier, depending upon the nature of the error.

The type 1 file contains an image of the system, if you skip the first record (which is information for SWTCH). However, there are minor complications due to the fact that the base page begins at location 2 and that it does not end at the end of a record in the file. The system main memory begins at the next record after the base page. The sample program below includes a subroutine (CALC) which takes this discrepancy into account. The program uses CALC to report the contents of the referenced memory location as it appears in the type 1 file and the record/word in the file. In order to actually patch the location, the program would have to be expanded or another program used to make the patch from the information printed. This sample is used only to illustrate the simplicity of patching.

```
FTN4,L
      PROGRAM LOOK (3,50),LOOK AT DISC FILE
C
C  RU,LOOK,TERMINAL,MEM LOC,NUM OF CONSECUTIVE LOCATIONS
C
      INTEGER LU(5),BUFR(128),DCB(144),NAME(3),SC,CR
      DATA NAME,SC,CR/2H!S,2HYS,2HTM,0,254/
      CALL RMPAR (LU)
      CALL OPEN (DCB,IERR,NAME,0,SC,CR)
      IF (IERR.LT.0) GO TO 700
      DO 100 I=LU(2),LU(2)+LU(3)-1
      CALL CALC (I,IREC,IWORD)
      CALL READF (DCB,IERR,BUFR,128,LEN,IREC)
      IF (IERR.LT.0) GO TO 700
100   WRITE (LU,1000) IREC,IWORD,BUFR(IWORD),BUFR(IWORD)
1000  FORMAT (" REC/WORD= ",I4,"/",I3," VALUE = ",@6,3X,A2)
      CALL CLOSE (DCB,IERR)
      CALL EXEC (6)
700   WRITE (LU,1010) IERR
1010  FORMAT (" FMP ERROR :"I6)
      END


      SUBROUTINE CALC (IMEM,IREC,IWORD)
      LOC = IMEM +126
      IF (IMEM.GT.1777B) LOC = LOC + 2
      IREC = (LOC)/128 + 2
      IWORD = MOD (LOC,128) + 1
      END
```

DOCUMENTATION

The following tables list currently available customer manuals for Data Systems Division products. This list supersedes the list in the last issue of the COMMUNICATOR 1000.

The most recent changes to the tables are indicated for easy reference. Prices are subject to change without notice.

Copies of manuals can be obtained from your local Sales and Service office. The address and telephone number of the office nearest to you are listed in the back of all customers manuals.

Customers in the U.S. may also order directly by mail. Simply list the name and part numnber of the manual(s) you need on the Corporate Parts Center form supplied at the back of the COMMUNICATOR 1000.

Change notices are free of charge. If you require a change notice only, send your request to:

Software/Publications Distribution
11000 Wolfe Road
Cupertino, CA. 95014

A few words about documentation terms:

*N A new manual refers only to the first printing of a manual. When first printed, a manual is assigned a part number.

*R A revised manual is a printing of an existing manual which incorporates new and/or changed information in its contents. For example, a manual is revised when a change notice is incorporated into the manual: the manual gets a new print date and the change notice disappears. Note that a revision to a manual obsoletes the previous version of the manual.

Change Notice A change notice is a supplement to an existing manual which contains new and/or changed information. It is issued when information must get to customers, yet it is inappropriate to issue a revised manual. A change notice has no part number; it is automatically included when you order the manual with which it is associated.

1000 SYSTEM MANUALS

PART NUMBER	MANUAL TITLE	PRICE	PRINT DATE	CHANGE NOTICE
02170-90006	HP 1000 Computer System Installation and Service	$ 2.50	7/77	
02172-90005	Getting Started with Your HP 1000 Disc Based Computer System (for A computers)	4.00	6/77	
02172-90010	Getting Started with Your HP 1000 Disc Based Computer System (for B computers)	2.50	3/78*R	
02173-90007	Getting Started with Your HP 1000 System: Models 20 and 21	7.00	8/77	
91780-93001	RJE/1000 Programming Manual	9.50	11/76	6/77

RTE SYSTEMS MANUALS

PART NUMBER	MANUAL TITLE	PRICE	PRINT DATE	CHANGE NOTICE
02313-93002	RTE 2313B Analog-Digital Interface Subsystem Operating and Service Manual	$30.00	8/76	12/77
02320-93002	RTE System Driver DVR76 for HP 2320A Low Speed Data Acquisition Subsystem Programming and Operating Manual	1.00	8/74	
02321-93001	RTE System Driver DVR74 for HP 2321A Low Speed Data Acquisition Subsystem Programming and Operating Manual	2.00	8/74	
09600-93010	RTE System DVR11 for HP 2892A Card Reader Programming and Operating Manual	1.00	8/74	
09600-93015	91200B TV Interface Kit: Programming and Operating Manual	4.50	7/75	1/76
09601-93005	RTE System Subroutine for General Purpose Registers	3.00	10/74	10/77
09601-93007	RTE Device Subroutine for HP 5327A/B-H48 Counter	2.50	12/74	
09601-93009	RTE Device Subroutine for HP 5326A-H18 Counter	2.50	12/74	
09601-93015	RTE for 40-bit Output Register #12556B	1.00	10/74	
09601-93017	RTE System Subroutine for HP 12555B D-A Converter	1.00	10/74	10/77
09603-93001	9603A/9604A Control System and Scientific Measurement Operating and Service Manual	7.50	5/76	
09610-93003	ISA FORTRAN Extension Package Reference Manual	4.50	12/77	
09611-90009	9611A Operating 406 Industrial Measurement and Control System	.25	4/75	
09611-90010	HP 6940A/B Multiprogrammer Verification Manual	4.50	8/75	
12604-93002	RTE DVR40 for 12604B Data Source Interface	1.00	8/74	
12665-93001	RTE System Driver DVR65 for HP 12771A Computer Serial Interface Kit	1.00	8/74	1/78
12732-90001	RTE Driver DVR33 Programming Manual	2.00	2/77	1/78
13197-90001	RTE Driver DVR36 Programming and Operating Manual	3.00	9/76	
24998-90001	DOS/RTE Relocatable Library Reference Manual	10.00	10/77	
25117-93003	RTE System Driver DVR24 for HP 7970 Series Digital Magnetic Tape Unit	1.00	8/74	
29003-93001	RTE System Driver DVR66 for HP 12772A Coupler Modem Interface Kit Programming and Operating Manual	1.00	8/74	
29003-93003	RTE System Driver DVR66 for HP 12770A Coupler Serial Interface Kit Programming and Operating Manual	1.00	8/74	
29009-93001	RTE System Driver62 for HP 2313B Subsystem	2.50	8/74	
29028-95001	RTE HP 2610A/2614A Line Printer Driver	1.50	8/73	
29029-95001	Real-Time Executive System Driver DVR00 for Multiple Device System Control Small Programs Manual	1.50	11/75	
29100-93001	RTE System Driver DVR40 (29100-60041) for HP 12604B Data Source Interface Programming and Operating Manual	1.00	8/76	
29101-93001	RTE Core-Based Software System Users Manual	10.00	1/76	
29102-93001	RTE BASIC Software System Programming and Operating Manual	10.00	3/74	8/75
29103-93001	RTE System Cross Loader Programming and Operating Manual	2.50	12/76	5/77
59310-90063	DVR37 Manual	3.50	6/77	
59310-90064	HP-IB Interface Bus I/O Kit Users Guide	8.50	4/77	6/77
91060-93005	RTE Driver for X-Y Display Storage Subsystem (HP Model 1331C-016) Programming and Operating Manual	1.00	8/74	
91062-93003	Real-Time Executive Driver for DVM/Scanner Subsystem	9.00	8/74	
91700-93001	Distributed System CCE Operating Manual	9.00	5/77	9/77
91705-93001	Distributed System SCE/5 Operating Manual	15.00	12/76	9/77
91200-90005	RTE Driver DVA13 for TV Interface (HP 91200B)	1.50	5/77	
91740-90002	DS/1000 Programmers Reference Manual	12.00	9/77*N	
91740-90015	DS/1000 A Guide for New Users	4.50	12/77*N	
92001-90010	RTE Line Printer Driver (DVA12) Reference Manual	1.00	11/77*R	
92001-90015	RTE DVR05 for 264X Terminals	2.00	1/78*R	
92001-93001	RTE-II Software System Programming and Operating Manual	10.00	7/77	10/77
92060-90004	RTE-III Software System Programming and Operating Manual	12.00	3/78*R	
92060-90005	RTE Assembler Reference Manual	7.00	11/77*R	
92060-90009	RTE-III General Information Manual	4.00	2/76	
92060-90010	RTE Batch/Spool Monitor and Operating System Pocket Guide	4.50	4/77	
92060-90012	RTE: A Guide for New Users	6.50	7/76	
92060-90013	Batch-Spool Monitor Reference Manual	9.50	10/77	
92060-90014	RTE Interactive Editor Reference Manual	6.00	5/77	
92060-90017	RTE Utility Programs	3.00	3/77	3/78
92060-90020	RTE On-Line Generator	15.00	2/78*R	
92062-90003	2631A/2635A Printer Utility Subroutine Reference Manual	1.50	1/78*N	
92400-93001	HP 92400A Utility Library Subroutines for Sensor Based Data Acquitition Systems Programming and Operating Manual	10.50	2/78*R	
92409-93001	Utility Library Subroutines for HP 7210A X-Y Plotter Programming and Reference Manual	2.50	12/77*N	

RTE SYSTEMS MANUALS (Continued)

PART NUMBER	MANUAL TITLE	PRICE	PRINT DATE	CHANGE NOTICE
92064-90002	RTE-M Programmer's Reference Manual	$14.00	1/78*R	2/78
92064-90003	RTE-M System Generation Reference Manual	7.50	1/78*R	2/78
92064-90004	RTE-M Editor Reference Manual	6.00	1/77	
92064-90007	RTE-M Pocket Guide	4.50	6/77	
92200-93001	RTE System Driver DVR12 for HP 2607A Line Printer Programming and Operating Manual	1.00	8/74	
92200-93005	Real-Time Executive Operating System Drivers and Device Subroutine Manual	5.00	1/78*R	
92202-93001	RTE System Driver DVR23 for HP 7970 Series Digital Mag Tape Units Programming and Operating Manual	1.00	8/74	
92400-93001	92400A Utility Library Subroutine for Sensor-Based Diagnostics	7.50	11/76	
93005-93005	Thermal Line Printer Subsystem for Driver DVR00 (RTE)	2.50	12/74	

HARDWARE MANUALS

PART NUMBER	MANUAL TITLE	PRICE	PRINT DATE	CHANGE NOTICE
02108-90002	HP 21MX M-Series Computer Reference Manual	$ 5.50	6/76	7/76
02108-90006	HP 21MX M-Series Computer Installation and Service Manual	10.00	7/76	
02108-90004	HP 21MX M-Series Computer Operators Manual	5.00	7/76	
02108-90017	21MX M-Series Computer Engineering and Reference Documentation	125.00	5/77	1/78
02108-90027	21MX-K-Series Computer Engineering and Reference Documentation	100.00	5/77	
02109-90001	HP 21MX E-Series Computer Operating and Reference Manual	8.00		
02109-90002	HP 21MX E-Series Computer Installation and Service Manual	15.00	8/76	3/77
02109-90006	HP 21MX M- and E-Series Computer I/O Interfacing Guide	7.00	10/77*R	12/77
02109-90014	21MX E-Series Computer HP 2109B and HP 2113B Operating and Reference Manual	8.00	8/77	
02109-90015	21MX E-Series Computer HP 2109B and HP 2113B Installation and Service Manual	15.00	8/77	9/77
12732-90005	HP 12732A 12733A Flexible Disc Subsystem Operating and Service Manual	5.50	8/77	
12979-90006	HP 12979A I/O Extender Installation and Service Manual	15.00	6/77	9/77
12979-90007	HP 12979A I/O Extender Operating and Reference Manual	5.00	12/75	9/77
12979-90014	HP 12979B Input Output Extender Operating and Reference Manual	2.00	8/77	
12979-90016	HP 12979B Input Output Extender Installation and Service Manual	12.00	8/77	8/77
12990-90003	HP 12990A Memory Extender Installation and Service Manual	5.50	4/76	8/76
5950-3765	21MX E-Series Computer Technical Reference Manual	3.50	6/77	

LANGUAGE MANUALS

PART NUMBER	MANUAL TITLE	PRICE	PRINT DATE	CHANGE NOTICE
02100-90140	Decimal String Arithmetic Routines	$ 6.50	2/77	
02108-90032	HP 21MX M-Series Computer RTE Microprogramming Reference Manual	15.00	10/76	9/77
02108-90034	HP 21MX M-Series Computer RTE Microprogramming Pocket Guide	2.75	1/77	
02109-90004	21MX E-Series Computer RTE Microprogramming Reference Manual	20.00	3/77	
02109-90008	21MX E-Series Computer RTE Microprogramming Pocket Guide	2.50	11/76	
02116-9014	HP Assembler Manual	6.50	8/75	
02116-9015	HP FORTRAN Manual	6.00	1/77	
02116-9016	Symbolic Editor	4.50	2/74	
02116-9072	ALGOL Reference Manual	10.00	11/76	
12907-90010	Implementing the HP 2100 Fast FORTRAN Processor	1.00	7/76	
24307-90014	DOS-III Assembler Reference Manual	8.00	7/74	
92060-90005	RTE Assembler Reference Manual	7.00	12/76	
92060-90016	Multi-User Real-Time BASIC Reference Manual	12.00	9/77	
92060-90023	RTE FORTRAN IV Reference Manual	10.00	7/77	
92063-90001	IMAGE 1000 Data Base Management System Reference Manual	9.00	10/77*R	12/77
92063-90004	IMAGE 1000 Data Base Management System Pocket Guide	4.00	6/77	
92065-90001	RTE-M Real-Time BASIC Language Reference Manual	8.50	2/77	7/77
02108-90008	HP 21MX M-Series Computer BCS and DOS Microprogramming Reference Manual	7.00	10/77*R	

SOFTWARE UPDATES

Following are cross-reference lists of the available 92001B, 92060B, and 92064A (options 20 & 40) software modules, the media on which the software modules are distributed, and the date code or revision of each module up to, and including level 1805.

NOTE:

For each module, interdependencies with other modules may exist (i.e., any updated module may require other updated modules to function properly).

SOFTWARE MODULE NUMBERS: 92001B LEVEL 1805 (RTE II)

The following modules are also available on a 7900 RTE Master Software Disc (#92001-13001), or a 7905 RTE Master Software Disc (#92001-13101).

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE
!S4LV7	24K SIO LINE PRINTER DRIVER	1538	92001-13305	[illegible]
%DVR15	RTE 7210A DRIVER	1738	92062-13304	[illegible]
%DVR33	FLEXIBLE DISC DRIVER	1805	92062-13304	12732-16001
!S4MT1	24K SIO MAG.TAPE DRIVER	1550	92001-13305	12971-16004
%DVR30	RTE FIXED HEAD DISC DRIVER	C	92062-13305	[illegible]
%CAL10	CAL. PLOTTER DRIVER	B	92062-13302	[illegible]
%CALIB	CAL. PLOTTER LIBRARY	C	92062-13302	[illegible]
%1FTN	FORTRAN MAIN CONTROL	E	92060-13308	[illegible]
%2FTN	FORTRAN PASS 1	E	92060-13308	[illegible]
%3FTN	FORTRAN PASS 2	E	92060-13308	[illegible]
%4FTN	FORTRAN PASS 3	E	92060-13308	[illegible]
%5FTN	FORTRAN PASS 4	E	92060-13308	[illegible]
%ALGOL	RTE/DOS ALGOL PART 1	1643	92062-13305	[illegible]
%ALGL1	RTE/DOS ALGOL PART 2	C	92060-13305	[illegible]
%FF.N	RTE/DOS FORMATTER	C	92060-13303	[illegible]
%DECAR	DOSN ST ARITH PK	A	92060-13303	[illegible]
%RLIB1	RTE/DOS LIBRARY PART 1	1740	92060-13302	[illegible]
%RLIB2	RTE/DOS LIBRARY PART 2	1740	92060-13302	[illegible]
%FF4.N	FORTRAN IV FORMATTER	1726	92060-13303	[illegible]
%DVR24	RTE 7970 7T MAG. TAPE DRIVER	D	92062-13305	[illegible]
%DVR31	RTE 7906A DISC DRIVER	1710	92062-13305	[illegible]
%DVR12	RTE 2767A DRIVER	A	92062-13303	[illegible]
%DVR00	RTE TTY/PUNCH/PHOTO READER	1740	92062-13302	[illegible]
%DVR11	RTE 2892A CARD READER DRIVER	1710	92062-13303	[illegible]
!S4LP	24K SIO LINE PRINTER	A	92001-13305	[illegible]
!S4SYD	24K SIO SYSTEM DUMP	A	92001-13305	[illegible]
!S4PHR	24K SIO PHOTO READER	A	92001-13305	[illegible]
!S4PUN	24K SIO TAPE PUNCH	A	92001-13305	[illegible]
!S4L67	24K SIO 2767 LINE PRINTER	A	92001-13305	[illegible]
!S4MT2	24K SIO 7970 MAG. TAPE	A	92001-13305	[illegible]
!S4MT3	24K SIO MAG. TAPE	A	92001-13305	[illegible]
!S4TER	24K SIO TERMINAL PRINTER	A	92001-13305	[illegible]
%1DV37	RTE HP-IB WITHOUT SRQ	1726	92062-13304	[illegible]
%2DV37	RTE HP-IB WITH SRQ	1726	92062-13304	[illegible]
%HPIB	HP-IB DEVICE SUBROUTINE	1710	92062-13304	[illegible]
%SRQ.P	SRQ.P TRAP UTILITY	1710	92062-13304	[illegible]
%1DV10	COMP. 7210A PLOTTER DRIVER	A	92062-13302	[illegible]
%2DV10	MIN. 7210A PLOTTER DRIVER	A	92062-13302	[illegible]
%DVA13	91200A DRIVER	1648	92062-13303	91200-16001
%TVLIB	91200A VIDEO MONITOR LIBRARY	1648	92062-13303	91200-16002
%LDR2	RTE II LOADER	1732	92001-13301	92001-16002
%TVVER	91200A TV INTERFACE VERIFIER	1648	92062-13303	[illegible]
%MTM	MULT. TERMINAL MONITOR	B	92060-13301	92001-16003
%SYLIB	RTE SYSTEM LIBRARY	1740	92060-13301	92001-16005
%AUTOR	AUTO RESTART PROGRAM	1631	92060-13310	92001-16014
%DVA12	2607/10/13/14/17/18 DRIVER	1805	92062-13303	92001-16020
%4DV05	RTE 2644/45 DRIVER	1805	92062-13302	92001-16027

(Continued)
SOFTWARE MODULE NUMBERS: 92001B LEVEL 1805 (RTE II)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE
12GN00	RTE-II 7900 OFF-LINE GEN.	1631	92001-13303	92001-16013
%AUTOR	AUTO RESTART PROGRAM	1631	92001-13302	92001-16014
12GNFH	RTE-II FIXED HEAD DISC GEN.	1631	92001-13306	92001-16018
%DVA12	2607/10/13/14/17/18 DRIVER	1805	92062-13303	92001-16020
12GN05	RTE-II 7905 OFF-LINE GEN.	1631	92001-13303	92001-16026
%4DV05	RTE 2644/45 DRIVER	1805	92062-13302	92001-16027
%0DV05	RTE 2640A DRIVER	1805	92062-13302	92001-16028
%$CMD2	RTE-II COMMAND PROGRAM	1710	92001-13301	92001-16029
%WHZT2	RTE-II WHZAT PROGRAM	1726	92001-13302	92001-16030
%RT2G1	RTE-II ON-LINE GENERATOR PT. 1	1805	92001-13304	92001-16031
%RT2G2	RTE-II ON-LINE GENERATOR PT. 1	1805	92001-13304	92001-16031
%DVA05	RTE DRIVER 264X MODEM	1740	92062-13302	92001-16035
&AUTOR	AUTO RESTART SOURCE	1631	92001-13302	92001-16014
&AN2F0	RTE-II 7900 GFATHER ANSW FILE	1805	92001-13307	92001-16033
&AN2F5	RTE-II 7905 GFATHER ANSW FILE	1805	92001-13307	92001-16034
%BMPG1	BATCH MONITOR PROGRAM PART 1	1631	92002-13301	92002-12001
%BMPG2	BATCH MONITOR PROGRAM PART 2	1631	92002-13301	92002-12001
%BMPG3	BATCH MONITOR PROGRAM PART 3	1631	92002-13301	92002-12001
42SP01	RTE-II SPOOL MONITOR PART 1	1631	92002-13303	92002-12002
%2SP02	RTE-II SPOOL MONITOR PART 2	1631	92002-13303	92002-12002
%BMLIB	BATCH LIBRARY	1631	92002-13302	92002-16006
%EDITR	RTE EDITOR	C	92002-13302	92002-16010
%ASMB	RTE ASSEMBLER	1634	92060-13304	92060-12004
%CLIB	RTE COMPILER LIBRARY	1726	92060-13315	92060-12005
%XREF	CROSS REFERENCE	A	92060-13304	92060-16028
4DVR32	RTE 7906A DISC DRIVER	A	92062-13305	92060-16031
%SWTCH	RTE-II SWITCH PROGRAM	1805	92001-13304	92060-16038
%SAVE	SAVE PROGRAM	1704	92060-13309	92060-16039
%RESTR	RESTORE PROGRAM	1704	92060-13309	92060-16040
%VERFY	DISC VERIFY PROGRAM	1704	92060-13309	92060-16041
%COPY	DISC COPY PROGRAM	1704	92060-13309	92060-16042
%DBKLB	DISC BACK UP LIBRARY	1704	92060-13309	92060-16043
1DSKUP	OFF LINE DISC BACK UP	1805	92060-13309	92060-16044
%RDNAM	READ NAME PROGRAM	1631	92001-13302	92060-16045
%KEYS	SOFT KEY UTILITY	1707	92001-13002	92060-16052
%KYDMP	SOFT KEY DUMP UTILITY	1707	92001-13002	92060-16053
%FTN4	RTE FORTRAN IV MAIN	1726	92060-13316	92060-16092
%FFTN4	RTE FORTRAN IV SEG F	1726	92060-13316	92060-16093
%0FTN4	RTE FORTRAN IV SEG 0	1726	92060-13316	92060-16094
%1FTN4	RTE FORTRAN IV SEG 1	1726	92060-13316	92060-16095
%2FTN4	RTE FORTRAN IV SEG 2	1726	92060-13316	92060-16096
%3FTN4	RTE FORTRAN IV SEG 3	1726	92060-13316	92060-16097
%4FTN4	RTE FORTRAN IV SEG 4	1726	92060-13316	92060-16098
&UPDAT	UPDATE TRANSFER FILE	1805	92001-13302	92060-16046
&PKDIS	PACK DISC TRANSFER FILE	1631	92001-13302	92060-16047
%LP31	SUBROUTINES FOR 2631/2635	1805		92062-16003
%MSAFD	FLEXIBLE DISC BACKUP UTILITY	1740	92060-13309	92064-16086
4DVR23	RTE 7970 9T. MAG. TAPE DRIVER	A	92062-13304	92202-16001
%2DV47	RTE 92900A DRIVER WITHOUT DMS	1643	92062-13302	92900-16002
%3DV47	RTE 92900A DRIVER WITH DMS	1631	92062-13302	92900-16003

SOFTWARE MODULE NUMBERS: 92060B LEVEL 1805 (RTE III)

The following modules are also available on a 7900 RTE Software Disc (#92060-13001), or a 7905 RTE Master Software Disc (#92060-13101), or a 7920 RTE Master Software Disc (#92060-13201).

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE
!S4LP7	24K SIO LINE PRINTER DRIVER	1538	92001-13305	92607-16004
%DVR15	RTE 7261A DRIVER	1738	92062-13304	92601-16021
%DVR33	FLEXIBLE DISC DRIVER	1805	92062-13304	12732-16001
!S4MT1	24K SIO MAG. TAPE DRIVER	1550	92001-13305	12970-16004
%DVR30	RTE FIXED HEAD DISC DRIVER	C	92062-13305	20747-60001
%CAL1B	CAL. PLOTTER DRIVER	B	92062-13302	20808-60001
%CAL1B	CAL. PLOTTER LIBRARY	C	92062-13302	20510-60001
%1FTN	FORTRAN MAIN CONTROL	E	92060-13308	20875-60001
%2FTN	FORTRAN PASS 1	E	92060-13308	20875-60002
%3FTN	FORTRAN PASS 2	E	92060-13308	20875-60003
%4FTN	FORTRAN PASS 3	E	92060-13308	20875-60004
%5FTN	FORTRAN PASS 4	E	92060-13308	20875-60005
%ALGOL	RTE/DOS ALGOL PART 1	1643	92060-13305	24129-60001
%ALGL1	RTE/DOS ALGOL PART 2	C	92060-13305	24129-60002
%FF.N	RTE/DOS FORMATTER	C	92060-13303	24153-60001
%DECAR	DOSM ST ARITH PK	A	92060-13303	24316-60001
%RLIB1	RTE/DOS LIBRARY PART 1	1740	92060-13302	24998-16001
%RLIB2	RTE/DOS LIBRARY PART 2	1740	92060-13302	24998-16001
%FF4.N	FORTRAN IV FORMATTER	1726	92060-13303	24998-16002
%DVR24	RTE 7970 7T MAG. TAPE DRIVER	D	92062-13305	25117-61499
%DVR31	RTE 7900A DISC DRIVER	1710	92062-13305	29013-60001
%DVR12	RTE 2767A DRIVER	A	92062-13303	29028-60002
%DVR00	RTE TTY/PUNCH/PHOTO READER	1740	92062-13302	29029-60001
%DVR11	RTE 2892A CARD READER DRIVER	1710	92062-13303	29030-60001
!S4LP	24K SIO LINE PRINTER	A	92001-13305	29100-60017
!S4SYD	24K SIO SYSTEM DUMP	A	92001-13305	29100-60018
!S4PHR	24K SIO PHOTO READER	A	92001-13305	29100-60019
!S4PUN	24K SIO TAPE PUNCH	A	92001-13305	29100-60020
!S4L67	24K SIO 2767 LINE PRINTER	A	92001-13305	29100-60022
!S4MT2	24K SIO 7970 MAG.TAPE	A	92001-13305	29100-60023
!S4MT3	24K SIO MAG. TAPE	A	92001-13305	29100-60049
!S4TER	24K SIO TERMINAL PRINTER	A	92001-13305	29100-60050
%1DV37	RTE HP-IB WITHOUT SRQ	1726	92062-13304	59310-16002
%2DV37	RTE HP-IB WITH SRQ	1726	92062-13304	59310-16003
%HPIB	HP-IB DEVICE SUBROUTINE	1710	92062-13304	59310-16004
%SRQ.P	SRQ.P TRAP UTILITY	1710	92062-13304	59310-16005
%1DV10	COMP. 7210A PLOTTER DRIVER	A	92062-13302	72008-60001
%2DV10	MIN. 7210A PLOTTER DRIVER	A	92062-13302	72009-60001
%DVA13	91200A DRIVER	1648	92062-13303	91200-16001
%TVLIB	91200A VIDEO MONITOR LIBRARY	1648	92062-13303	91200-16002
%TVVER	91200A TV INTERFACE VERIFIER	1648	92062-13303	91200-16004
%MTM	MULT. TERMINAL MONITOR	B	92001-13301	92001-16003
%2DP43	POWER FAILURE DRIVER	1633	92001-13301	92001-16004
%SYLIB	RTE SYSTEM LIBRARY	1740	92001-13301	92001-16005
%CR2SY	CORE RESIDENT OPERATING SYS.	1740	92001-13301	92001-16012

(Continued)
SOFTWARE MODULE NUMBERS: 92060B LEVEL 1805 (RTE III)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE
%0DV05	RTE 2640A DRIVER	1805	92062-13302	92001-16028
%DVA05	RTE DRIVER 264X MODEM	1805	92062-13302	92001-16035
%AUTOR	AUTO RESTART PROGRAM SOURCE	1631	92060-13310	92001-16014
%BMPG1	BATCH MONITOR PROGRAM PART 1	1631	92002-13301	92002-12001
%BMPG2	BATCH MONITOR PROGRAM PART 2	1631	92002-13301	92002-12001
%BMPG3	BATCH MONITOR PROGRAM PART 3	1631	92002-13301	92002-12001
%BMLIB	BATCH LIBRARY	1631	92002-13302	92002-16006
%EDITR	RTE EDITOR	C	92002-13302	92002-16010
%3SP01	RTE-III SPOOL MONITOR PART 1	1631	92060-13313	92060-12001
%3SP02	RTE-III SPOOL MONITOR PART 2	1631	92060-13313	92060-12001
%CF3SY	MEMORY RESIDENT SYSTEM	1740	92060-13301	92060-12003
%ASMB	RTE ASSEMBLER	1639	92060-13304	92060-12004
%CLIB	RTE COMPILER LIBRARY	1726	92060-13315	92060-12005
%3DF43	POWER FAILURE DRIVER	1633	92060-13301	92060-16001
%LDR3	RTE-III LOADER	1732	92060-13301	92060-16004
%WHAT3	RTE-III WHZAT PROGRAM	1732	92060-13310	92060-16006
%XREF	CROSS REFERENCE	A	92060-13304	92060-16028
!3GN00	7900 RTE-III GENERATOR	1631	92060-13311	92060-16029
%DVR32	RTE 7905A DISC DRIVER	A	92062-13305	92060-16031
!3GN05	7905 RTE-III GENERATOR	1631	92060-13311	92060-16032
%$PVMP	$PVMP	A	92060-13301	92060-16035
%$CMD3	RTE-III COMMAND PROGRAM	1710	92060-13301	92060-16036
%RT3G1	RTE-III ON-LINE GENERATOR PT.1	1805	92060-13312	92060-16037
%RT3G2	RTE-III ON-LINE GENERATOR PT.2	1805	92060-13312	92060-16037
%SWTCH	RTE-III SWITCH PROGRAM	1805	92060-13312	92060-16038
%SAVE	SAVE PROGRAM	1704	92060-13309	92060-16039
%RESTR	RESTORE PROGRAM (RSTOR)	1704	92060-13309	92060-16040
%VERFY	DISC VERIFY PROGRAM	1704	92060-13309	92060-16041
%COPY	DISC COPY PROGRAM	1704	92060-13309	92060-16042
%DBKLB	DISK BACK UP LIBRARY	1704	92060-13309	92060-16043
!DSKUP	OFF LINE DISK PACK UP	1805	92060-13309	92060-16044
%RDNAM	READ NAME PROGRAM	1631	92060-13310	92060-16045
%KEYS	SOFT KEY UTILITY	1707	92062-13310	92060-16052
%KYDMP	SOFT KEY DUMP UTILITY	1707	92060-13310	92060-16053
%FTN4	RTE FORTRAN IV MAIN	1726	92060-13316	92060-16092
%FFTN4	FORTRAN IV SEGMENT F	1726	92060-13316	92060-16093
%0FTN4	FORTRAN IV SEGMENT 0	1726	92060-13316	92060-16094
%1FTN4	FORTRAN IV SEGMENT 1	1726	92060-13316	92060-16095
%2FTN4	FORTRAN IV SEGMENT 2	1726	92060-13316	92060-16096
%3FTN4	FORTRAN IV SEGMENT 3	1726	92060-13316	92060-16097
%4FTN4	FORTRAN IV SEGMENT 4	1726	92060-13316	92060-16098
%UPDAT	UPDATE TRANSFER FILE	1805	92060-13310	92060-16046
%PKDIS	PACK DISK TRANSFER FILE	1631	92060-13310	92060-16047
%AN3F0	RTE-III 7900 GATHER ANSW FILE	1805	92060-13314	92060-16050
%AN3F5	RTE-III 7905/20 GATHER ANS FILE	1805	92060-13314	92060-16051
%MSAFD	FLEXIBLE DISC BACKUP UTILITY	1740	92060-13309	92060-16086
%DVR23	RTE 7970 9T. MAG. TAPE DRIVER	A	92062-13304	92002-16001
%2DV47	RTE 9895A DRIVER WITHOUT DMS	1726	92062-13302	92002-16002
%3DV47	RTE 9895A DRIVER WITH DMS	1643	92062-13302	92002-16003

SOFTWARE MODULE NUMBERS: 92062A LEVEL 1805 (RTE III)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE
%DVR15	RTE 7261A DRIVER	1738	92062-13304	09601-16021
%DVR33	FLEXIBLE DISC DRIVER	1805	92062-13304	12732-16001
%DVR30	RTE FIXED HEAD DISC DRIVER	C	92062-13305	20747-60001
%CAL1M	CAL. PLOTTER DRIVER	B	92062-13302	20608-60001
%CALIB	CAL. PLOTTER LIBRARY	C	92062-13302	20810-60001
%DVR24	RTE 7970 7T MAG. TAPE DRIVER	D	92062-13305	25117-60499
%DVR31	RTE 7900A DISC DRIVER	1710	92062-13305	29013-60001
%DVR12	RTE 2767A DRIVER	A	92062-13303	29026-60002
%DVR00	RTE TTY/PUNCH/PHOTO READER	1740	92062-13302	29029-60001
%DVR11	RTE 2892A CARD READER DRIVER	1710	92062-13303	29030-60001
%1DV37	RTE HP-IB WITHOUT SRQ	1726	92062-13304	59310-16002
%2DV37	RTE HP-IB WITH SRQ	1726	92062-13304	59310-16003
%HPIB	HP-IB DEVICE SUBROUTINE	1710	92062-13304	59310-16004
%SRQ.P	SRQ.P TRAP UTILITY	1710	92062-13304	59310-16005
%1DV10	COMP. 7210A PLOTTER DRIVER	A	92062-13302	72008-60001
%2DV10	MIN. COMP. 7910A PLOTER DRIVE	A	92062-13302	72009-60001
%DVA13	9120HA DRIVER	1648	92062-13303	91200-16001
%TVLIB	9120DA VIDEO MONITOR LIBRARY	1648	92062-13303	91200-16002
%TVVER	9120HA TV INTERFACE VERIFIER	1648	92062-13303	91200-16004
%DVA12	2607/10/13/14/17/18 DRIVER	1805	92062-13303	92001-16020
%4DV05	RTE 2644/45 DRIVER	1805	92062-13302	92001-16027
%0DV05	RTE 2640A DRIVER	1805	92062-13302	92001-16028
%DVA05	RTE DRIVER 264X MODEM	1740	92062-13302	92001-16035
%DVR32	RTE 7905A DISC DRIVER	A	92062-13305	92060-16031
%DVR23	RTE 7970 9T. MAG. TAPE DRIVER	A	92062-13304	92202-16001
%2DV47	RTE 9290HA DRIVER WITHOUT DMS	1643	92062-13302	92900-16002
%3DV47	RTE 9290HA DRIVER WITH DMS	1643	92062-13302	92900-16003

SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1805 (RTE-M)

The following modules are unique in that they are available on Flexible disc as well as Paper Tape and Mini-Cartridge.

STRUCTURE

The RTE-M operating system is divided into three groups. Refer to the RTE-M Programmer's Reference Manual (part no. 92064-90002) for a description of the operating systems.

Within this list the modules that correspond with each operating system are described as MI, MII, or MIII.

CARTRIDGE TAPES

There are three cartridge tapes that contain the three operating systems. The part numbers of these cartridge tapes and the corresponding operating systems follow:

92064-13301	RTE-MI
92064-13302	RTE-MII
92064-13303	RTE-MIII

Modules that correspond with two or all three operating systems and are contained on more than one cartridge tape contain (MI), (MII), or (MII) in their description.

Modules that do not directly relate to the operating systems are contained on the other cartridge tapes.

FLEXIBLE DISCS

There are two flexible discs referred to as GEN DISC and APP DISC. The GEN DISC (92064-13401) contains all the software that can be loaded at generation. The APP DISC (92064-13402) contains all the application software that can be loaded on-line. As with the cartridge tapes, some of the modules can be found on both flexible discs.

The Generation disc contains the following:

- Off-line generator
- All operating system software
- I/O drivers
- Certain HP user programs

The Applications disc contains the following:

- HP applications programs — Assembler
  FORTRAN compiler
  Editor
  Cross reference program
- Certain relocatable system software
- Certain user programs

Modules that appear on both flexible discs contain (GEN DISC) or (APP DISC) in their description.

SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1805 (RTE-M)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE	FLEXIBLE DISC
%DVR15	RTE 7261A CARD READER DRIVER	1738	92062-13304	09601-16021	92064-13401
%DVR33	FLEXIBLE DISC DRIVER	1805	92062-13304	12732-16001	92064-13401
%CAL10	RTE PLOTTER DRIVER	B	92062-13302	20806-60001	92064-13401
%CALIB	CAL. PLOTTER LIBRARY	C	92062-13302	20810-60001	92064-13401
%FF.N	RTE/DOS FORTRAN FORMATTER	C	92060-13303	24153-60001	92064-13402
%FF.N	RTE/DOS FORTRAN FORMATTER	C	92060-13303	24153-60001	92064-13401
%DECAR	DOSM STRING ARITH PK	A	92060-13303	24306-60001	
%RLIB1	RTE/DOS LIBRARY	1740	92060-13302	24998-16001	92064-13401
%RLIB1	RTE/DOS LIBRARY	1740	92060-13302	24998-16001	92064-13402
%RLIB2	RTE/DOS LIBRARY	1740	92060-13302	24998-16001	92064-13402
%RLIB2	RTE/DOS LIBRARY	1740	92060-13302	24998-16001	92064-13401
%FF4.N	FORTRAN IV FORMATTER	1624	92060-13303	24998-16002	92064-13402
%FF4.N	FORTRAN IV FORMATTER	1624	92060-13303	24998-16002	92064-13401
%DVR12	RTE 2767A DRIVER	A	92062-13303	29028-60002	92064-13401
%DVR00	RTE TTY/PUNCH/PHOTO READER	1740	92062-13302	29029-60001	92064-13401
%DVR11	RTE 2892A CARD READER DRIVER	1710	92062-13303	29030-60001	92064-13401
%1DV37	HP-IB WITHOUT SYSTEM REQUEST	1710	92062-13304	59310-16002	92064-13401
%2DV37	HP-IB WITH SYSTEM REQUEST	1710	92062-13304	59310-16003	92064-13401
%HPIB	HP-IB RTE UTILITY	1710	92062-13304	59310-16004	92064-13401
%SRQ.P	SRQ.P TRAP UTILITY	1710	92062-13304	59310-16005	92064-13401
%1DV10	COMP. 7210A PLOTTER DRIVER	A	92062-13302	72008-60001	92064-13401
%2DV10	MIN. COMP. 7210A PLOTTER DRIVE	A	92062-13302	72009-60001	92064-13401
%DVA13	91200 TV INTERFACE DRIVER	1648	92062-13303	91200-16001	92064-13401
%TVLIB	VIDEO MONITOR LIBRARY	1648	92062-13303	91200-16002	92064-13401
%TVVER	TV INPT VERIF	1648	92062-13303	91200-16004	92064-13401
%DVA12	2607/10/13/14/17/18 DRIVER	1805	92062-13303	92001-16020	92064-13401
%4DV05	RTE 2644/45 DRIVER	1805	92062-13302	92001-16027	92064-13401
%0DV05	RTE 2640A DRIVER	1805	92062-13302	92001-16028	92064-13401
%DVA05	RTE DRIVER 264X MODEM	1740	92062-13302	92001-16035	92064-13401
%KEYS	SOFT KEY UTILITY	1707	92064-13304	92060-16052	92064-13402
%KYDMP	SOFT KEY DUMP UTILITY	1707	92064-13304	92060-16053	92064-13402
%FTN4	FORTRAN IV MAIN	1726		92060-16092	92064-13402
%FFTN4	RTE FORTRAN IV SEG ID SUB	1726		92060-16093	92064-13402
%0FTN4	FORTRAN IV SEGMENT 0	1726		92060-16094	92064-13402
%1FTN4	FORTRAN IV SEGMENT 1	1726		92060-16095	92064-13402
%2FTN4	FORTRAN IV SEGMENT 2	1726		92060-16096	92064-13402
%3FTN4	FORTRAN IV SEGMENT 3	1726		92060-16097	92064-13402
%4FTN4	FORTRAN IV SEGMENT 4	1726		92060-16098	92064-13402
%FMPC	CARTRIDGE FMP/FMPCR (LIB)	1805	92064-13306	92064-12005	92064-13401
%FMPF	FLEX DISC FMGR LIB (GEN DISC)	1805		92064-12006	92064-13401
%FMPF	FLEX DISC FMGR LIB (APP DISC)	1726		92064-12006	92064-13402
%CLIBM	RTE COMPILER LIBRARY	1726		92064-12007	92064-13402
%MSY1	MI OPERATING SYSTEM	1805	92064-13301	92064-16001	92064-13401
%MSY2	MII OPERATING SYSTEM	1805	92064-13302	92064-16002	92064-13401
%MSY3	MIII OPERATING SYSTEM	1805	92064-13303	92064-16003	92064-13401
%MBU	MI BUFFERING	1650	92064-13301	92064-16005	92064-13401

(Continued)
SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1805 (RTE-M)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE	FLEXIBLE DISC
%MMP	MI SCHEDULING OPTION	1805	92064-13301	92064-16006	92064-13401
%MTI	TIMER OPTION (MII)	1650	92064-13302	92064-16008	92064-13401
%MTI	TIMER OPTION (MIII)	1650	92064-13303	92064-16008	92064-13401
%MTI	TIMER OPTION (MI)	1650	92064-13301	92064-16008	92064-13401
%MTS	TIME SCHEDULING OPTION (MIII)	1650	92064-13303	92064-16009	92064-13401
%MTS	TIME SCHEDULING OPTION (MII)	1650	92064-13302	92064-16009	92064-13401
%MTS	TIME SCHEDULING OPTION (MI)	1650	92064-13301	92064-16009	92064-13401
%MOP	OPERATOR COMMAND OPTION (MIII)	1650	92064-13303	92064-16010	92064-13401
%MOP	OPERATOR COMMAND OPTION (MII)	1650	92064-13302	92064-16010	92064-13401
%MOP	OPERATOR COMMAND OPTION (MI)	1650	92064-13301	92064-16010	92064-13401
%MCL	CLASS I/O OPTION (MII)	1808	92064-13302	92064-16011	92064-13401
%MAP	MI/II ABSOLUTE PROGRAM LOADER	1726	92064-13305	92064-16012	92064-13401
%MDMLB	DUMMY LIBRARY (MII)	1650	92064-13302	92064-16013	92064-13401
%MDMLB	DUMMY LIBRARY (MI)	1650	92064-13301	92064-16013	92064-13401
%MDMLB	DUMMY LIBRARY (MIII)	1650	92064-13303	92064-16013	92064-13401
%MCL3	CLASS I/O OPTION (MIII)	1808	92064-13303	92064-16015	92064-13401
%MAP3	MIII ABSOLUTE PROGRAM LOADER	1726	92064-13305	92064-16016	92064-13401
%FMGC0	CARTRIDGE FILE MANAGER	1805	92064-13305	92064-16017	92064-13401
%DRC	CARTRIDGE DIR MAN PROGRAM	1650	92064-13304	92064-16018	92064-13401
%TBLCR	CARTRIDGE DIRECTORY TABLES	1650	92064-13304	92064-16019	92064-13401
%DRC1	MI CARTRIDGE DIRECTORY SUBR	1650	92064-13306	92064-16021	92064-13401
%RTMGN	SYSTEM GENERATOR	1726	92064-13305	92064-16022	92064-13401
%RTMLD	RELOCATING LOADER (GEN DISC)	1726	92064-13305	92064-16023	92064-13401
%RTMLD	RELOCATING LOADER (APP DISC)	1726	92064-13305	92064-16023	92064-13402
%RTMSC	LOADER SUB CONTROL (APP DISC)	1805	92064-13305	92064-16024	92064-13402
%RTMSC	LOADER SUB CONTROL (GEN DISC)	1805	92064-13305	92064-16024	92064-13401
%MEDIT	EDITOR	1703		92064-16025	92064-13402
%MASM6	CROSS REFERENCE SEGMENT	1805		92064-16026	92064-13402
%MPF	MI/II POWER FAIL	1650	92064-13304	92064-16027	92064-13401
%MPF3	MIII POWER FAIL	1650	92064-13304	92064-16029	92064-13401
%MAUTO	AUTOR REL	1650	92064-13304	92064-16030	92064-13401
%MRN	RESOURCE NUMBER MNGR (MIII)	1650	92064-13303	92064-16031	92064-13401
%MRN	RESOURCE NUMBER MANAGER (MII)	1650	92064-13302	92064-16031	92064-13401
%ONMTM	MULTI TERMINAL MONITOR (APP D)	1650	92064-13305	92064-16032	92064-13402
%ONMTM	MULTI TERMINAL MONITOR (GEN D)	1650	92064-13305	92064-16032	92064-13401
!MCGEN	ABSOLUTE CARTRIDGE GENERATOR	1805	92064-13307	92064-16033	
%SGPRP	SEGMENT PROGRAM PREP	1650		92064-16034	92064-13402
%MPRMP	PROMPT (MTM)	1650	92064-13305	92064-16035	92064-13401
%MRSPN	RESPONSE (MTM)	1650	92064-13305	92064-16036	92064-13401
%MASM0	ASSEMBLER MAIN CONTROL	1805		92064-16040	92064-13402
%MASM1	ASSEMBLER SEGMENT 1	1650		92064-16041	92064-13402
%MASM2	ASSEMBLER SEGMENT 2	1650		92064-16042	92064-13402
%MASM3	ASSEMBLER SEGMENT 3	1650		92064-16043	92064-13402
%MASM4	ASSEMBLER SEGMENT 4	1650		92064-16044	92064-13402
%MFTN0	FORTRAN MAIN CONTROL	1650		92064-16045	92064-13402
%MFTN1	FORTRAN SEGMENT 1	1650		92064-16046	92064-13402

(Continued)
SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1805 (RTE-M)

MODULE	DESCRIPTION	REVISION CODE	MINI CARTRIDGE	PAPER TAPE	FLEXIBLE DISC
%MFNT2	FORTRAN SEGMENT 2	1650		92064-16047	92064-13402
%MASM5	ASSEMBLER SEGMENT 0	1650		92064-16050	92064-13402
%MXRF0	CROSS REFERENCE MAIN	1650		92064-16051	92064-13402
%DIRD	CARTRIDGE DIRECTORY READ	1650	92064-13304	92064-16054	92064-13401
%FMGF0	FLEX DISC FILE MNGR (GEN DISC)	1805		92064-16055	92064-13401
%FMGF0	FLEX DISC FILE MNGR (APP DISC)	1709		92064-16055	92064-13402
%DRF	F DISC DIRECT PROG (APP DISC)	1650		92064-16056	92064-13402
%DRF	F DISC DIRECT PROG (GEN DISC)	1650		92064-16056	92064-13401
%TBLFP	FLEXIBLE DISC DIRECT TABLES	1709		92064-16057	92064-13401
%DRF1	F DISC DIRECTORY SUB (APP D)	1650		92064-16060	92064-13402
%DRF1	F DISC DIRECTORY SUB (GEN D)	1650		92064-16060	92064-13401
IMFGEN	ABSOLUTE FLEXIBLE DISC SYSTEM	1805		92064-16075	92064-13401
%STRTM	RTE-M SYSTEM START-UP	1709	92064-13304	92064-16080	92064-13401
%MSYLB	RTE-M SYSTEM LIBRARY (GEN DISC	1709	92064-13306	92064-16081	92064-13401
%MSYLB	RTE-M SYSTEM LIBRARY (APP DISC	1709	92064-13306	92064-16081	92064-13402
%MSAFD	FLEXIBLE DISC BACKUP UTILITY	1740	92060-13309	92064-16086	92064-13402
&TBLCR	CARTRIDGE DIRECTORY TBLS SOURC	1650	92064-13306	92064-18059	92064-13402
&MHELP	EDITOR HELP FILE SOURCE	1650		92064-18126	92064-13402
&MAUTO	AUTOR SOURCE	1650	92064-13306	92064-18141	92064-13402
&TBLFP	FLEXIBLE DISC DIRECTORY SOURCE	1709		92064-18171	92064-13402
%DVR23	RTE 7970 9T. MAG. TAPE DRIVER	A	92062-13304	92202-16001	92064-13401
%2DV47	RTE 92900A DRIVER WITHOUT DMS	1643	92062-13302	92900-16002	92064-13401
%3DV47	RTE 92900A DRIVER WITH DMS	1643	92062-13302	92900-16003	92064-13401

TRAINING SCHEDULE

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

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

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

DATA SHEETS

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

REGISTRATION

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

ACCOMMODATIONS

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

CANCELLATIONS

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

NEW COURSES

The following new courses have been added to the HP 1000 Computer Systems Training Schedule since the last issue of the Communicator.

22952B HP 1000 ASSEMBLER
PROGRAMMING COURSE

Description: This course covers the operation of the RTE assembler in an HP 1000 computer system environment. Major emphasis is placed on the development of assembly language programs for use in an RTE operating system.

Length: 5 days.

Lab: Provides extensive hands-on experience in the coding, editing, assembly and debugging of RTE assembler programs using an HP 1000 system.

Prerequisites: Completion of either the RTE-II/III Operating Systems Course (22965B) or the RTE-M Operating System Course (22985A), or equivalent RTE experience.

22987A DS/1000 USER'S COURSE

Description: This course covers the fundamentals of the HP DS/1000 Distributed Systems Network, including: network philosophy, operator commands, remote I/O, remote file access, remote EXEC calls, program-to-program calls, and store-and-forward communications. Information is provided on both memory-based and disc-based RTE systems operation in addition to information on an HP 3000 MPE link.

Length: 5 days.

Lab: Provides hands-on experience on programming of a multi-node DS/1000 distributed systems network.

Prerequisites: Completion of either the RTE-II/III Operating Systems Course (22965B) or the RTE-M Operating System Course (22985A), or equivalent RTE experience. The HP 3000 Comprehensive Introduction Course (22801A) is also recommended for those customers whose networks include an HP 3000 node.

22961B THEORY OF OPERATION
OF DS/1000

Description: This course provides a thorough exposure to the internal functioning of the DS/1000 software as it relates to an HP 1000-to-HP 1000 link. Topics covered include communications management, microcoded driver, remote file manager, network configuration, link protocol, generation and performance evaluation. Information is provided on the level of program listings, flowcharts, and tables.

Note: Customers whose networks include an HP 3000 node should also take the one-day DS/1000 to HP 3000 Theory of Operation Course (22962B).

Length: 4 days.

Lab: Provides hands-on programming of a DS/1000 network, use of system utilities, diagnostics, and troubleshooting tools. System generation and network configuration are covered in detail.

Prerequisites: Completion of the DS/1000 User's Course (22987A).

22962B THEORY OF OPERATION FOR DS/1000-TO-HP 3000

Description: This course provides a thorough exposure to the internal functioning of the DS/1000 software as it relates to an HP 1000-to-HP 3000 link. Topics covered include communications management, network configuration, link protocol, HP 1000 as a master to MPE, and HP 1000 as a slave to MPE. Information is provided on the level of program listings, flowcharts, and tables.

Length: 1 day.

Lab: None.

Prerequisites: Completion of the Theory of Operation of DS/1000 Course (22961B), which is normally taken earlier in the same week.

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

Course Number	Title	Length	Price	CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Sunnyvale)	Boise Division (Boise)
22965B	RTE-II-III (Course includes RTE-II/III operating system, tch spool monitor and file manager.)	10 days	1000	May 15 June 5 Jun 19 Jul 10 Jul 24 Aug 7 Aug 21	May 1 Jun 5 Jul 17 Aug 14	May 1 May 15 Jun 12 Jul 10 Jul 24 Aug 7 Aug 21				
22985A	RTE-M	5 days	500	Jun 26 Aug 14		Jun 26				
22977A*	IMAGE	5 days	500	May 15 Jul 24	May 15 Jun 19 Jul 31	Jun 5 Aug 21				
22952B*	1000 ASMB	5 days	500	Jun 26 Aug 14		Jun 5 Jul 17 Aug 28				
22987A*	DS/1000 User's Course	5 days	500	May 8 Jun 5 Jul 10 Aug 21		Jul 10				
22961B*	DS/1000 Theory of Op.	4 days	400	Jun 12 Aug 28		Jul 17				
22962B*	DS/1000 HP 3000 Theory of Op.	1 day	100	Jun 16 Sep 1		Jul 21				

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

Course Number	Title	Length	Price	CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Sunnyvale)	Boise Division (Boise)
22990A*	RTE-Driver Writing	3 days	300	May 22 Jul 31		Jul 5				
22980B*	HP-IB Minicomputer Environment	4 days	400	Jun 19 Aug 21						
22983A*	21MX-E Microprogram-ming	5 days	500	Jul 17						
92780A*	HP-ATS Automatic Test System	5 days	1000				May 8 Jun 5 Jul 10			
13294A	Dev. Terminal	5 days	500					Jul 10		
22940A	21 Maint.	10 days	1000						May 15 Jul 10 Aug 7	
22941A	21MX Maint.	5 days	500						May 1 Jun 5 Jun 26 Jul 24 Jul 31 Aug 21 Aug 28	
22942A	7900 Maint.	5 days	500						Jun 12 Jul 31 Aug 28	
22945A	7905 Maint.	5 days	500						May 8 Jun 19 Jul 17 Aug 7	

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

Course Number	Title		CUPERTINO Customer Training Center	FULLERTON Customer Training Center	ROCKVILLE Customer Training Center	Data Systems Division (Cupertino)	Data Terminals Division (Cupertino)	Customer Service Division (Sunnyvale)	Boise Division (Boise)
	Length	Price							
91302A	2645 Maint.							May 31	
	3 days	300							
22943A	7970B Maint.								
	5 days	600							
22944A	7970E Maint.								
	5 days	600							

**22965B RTE-II/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.*

U.S. TRAINING CENTER ADDRESSES

Cupertino

CUSTOMER TRAINING CENTER
19310 Pruneridge Avenue
Cupertino, CA 95014
(408) 996-9800

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

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

Fullerton

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

Rockville

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

Sunnyvale

CUSTOMER SERVICE DIVISION
974 East Arques Avenue
Sunnyvale, CA 94086
(408) 735-1550

Boise

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

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS

Course Number	Title / Length	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
22965B	RTE-II/III 10 days (Course includes RTE-II/III operating system, batch spool monitor and file manager.)	May 29 Jun 19 Jul 31 Aug 28 Sept 25 Oct 23	June 19 Oct 9 Dec 4	Jun 19 Oct 23	May 8 Jun 19 Jul 24 Sep 4	May 29 (R) Jul 3 (M) Oct 9 (M)	May 22 Aug 28 Oct 9 Nov 27		Jun 5 Jul 17 Aug 28 Nov 6	Jun 26 Aug 7 Oct 9	Jun 5 Oct 2
22985A	RTE-M 5 days	Jul 10 Sep 18									
22977A*	IMAGE 5 days	Jul 17 Sep 4		Jul 3 Nov 6	May 30 Aug 29 Oct 23				May 22 Jul 3 Sep 25	Oct 23	
22952B*	1000 ASMB 5 days	May 29 Jul 24 Sep 11 Oct 23	Sep 11 Nov 6	Jun 12 Oct 16	Jun 5 Jul 10 Aug 14 Oct 16	May 15 (R) Jun 19 (M) Sep 11 (M)	Jun 5 Sep 11 Dec 11		Jun 19 Sep 18		
22987A*	DS/1000 User's Course 5 days	Jul 3 Oct 9								Jul 10	
22961B*	DS/1000 4 days	Aug 28								Jul 17	
22962B*	DS/1000 HP 3000 Theory of Op. 1 day	Sep 1								Jul 21	
22990A*	RTE Driver Writing 3 days										
22980B*	HP-IB Minicomputer Environment 4 days	May 15 Aug 14									
22983A*	21MX-E Microprogramming 5 days							May 22			

EUROPEAN TRAINING CENTER SCHEDULES AND LOCATIONS (Continued)

Course Number	Title / Length	Boblingen	Amsterdam	Madrid	Winnersh	Milan (M) Rome (R)	Stockholm	Grenoble	Orsay	Vienna	Brussels
92780A*	HP-ATS Automatic Test System / 5 days										
13294A	Dev. Terminal / 5 days										
22940A	2100 Maint. / 10 days										
22941A	21MX Maint. / 5 days							May 22			
22942A	7900 Maint. / 5 days										
22945A	7905 Maint. / 5 days							May 15 May 22			
22945A	7905 Maint. / 5 days										
91302A	2645 Maint. / 3 days										
22943A	7970B Maint. / 5 days										
22944A	7970E Maint.										
40270A	Intro to HP Computers / 5 days	Jun 5 Jul 17 Sep 25							May 8 Jul 31 Oct 9		
22965B-H01	FORTRAN IV / 5 days	Jun 12 Oct 16		Jun 5 Oct 2							

**22965B RTE-II/III is a prerequisite for these courses. Other prerequisites may also apply – refer to the data sheet for each course for more information.*

EUROPEAN TRAINING CENTER ADDRESSES

Boblingen

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

Brussels

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

Stockholm

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

Madrid

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

Amsterdam

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

Orsay

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

Grenoble

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

Vienna

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

Milan

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

Winnersh

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

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name _____ **Title** _____

Company _____

Street _____

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

Country _____

☐ HP Employee Account Number _____ Location Code _____

☐ **DIRECT SUBSCRIPTION**

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

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

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

☐ **SERVICE CONTRACT CUSTOMERS**

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

Number of additional copies _____

FOR HP USE ONLY

CONTRACT KEY

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

Approved _____

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

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

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

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

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

SAMPLE

☒ **DIRECT SUBSCRIPTION**

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

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

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

SAMPLE

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

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

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

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

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

HEWLETT-PACKARD
COMPUTER SYSTEMS COMMUNICATOR ORDER FORM

Please Print:

Name _____ **Title** _____

Company _____

Street _____

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

Country _____

☐ HP Employee Account Number _____ Location Code _____

☐ **DIRECT SUBSCRIPTION**

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

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

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

☐ **SERVICE CONTRACT CUSTOMERS**

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

Number of additional copies _____

FOR HP USE ONLY

CONTRACT KEY

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

Approved _____

HEWLETT-PACKARD
COMMUNICATOR SUBSCRIPTION AND ORDER INFORMATION

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

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

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

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

SAMPLE

☒ **DIRECT SUBSCRIPTION**

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

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

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

SAMPLE

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

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

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

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

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

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

HEWLETT hp PACKARD

CONTRIBUTED SOFTWARE
Direct Mail Order Form

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

Please Print:

Name _____ **Title** _____

Company _____

Street _____

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

Country _____

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

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

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

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

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

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

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

ORDERING INFORMATION

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

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

For example:

22113A-K01

The different options are:

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

NOTE

Specify 800 BPI or 1600 BPI Magnetic tape.

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

Not all options are available for all programs.

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

For example:

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

Only those options listed in each abstract are available.

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

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

HEWLETT-PACKARD
LOCUS CONTRIBUTED SOFTWARE CATALOG
DIRECT MAIL ORDER FORM

Please Print:

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

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

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

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

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

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

NOT TO BE USED FOR ORDERING COMMUNICATOR SUBSCRIPTIONS

HEWLETT hp PACKARD

CORPORATE PARTS CENTER

Direct Mail
Parts and Supplies Order Form

SHIP TO:

NAME _____

COMPANY _____ CUSTOMER REFERENCE # _____

STREET _____ TAXABLE*? _____

CITY _____ STATE _____ ZIP CODE _____

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

Special Instructions

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

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

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

When completed, please mail this form with payment to:

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

Phone: (415) 968-9200

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

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

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

Printed in U.S.A. 4/78

Part No. 5951-6111