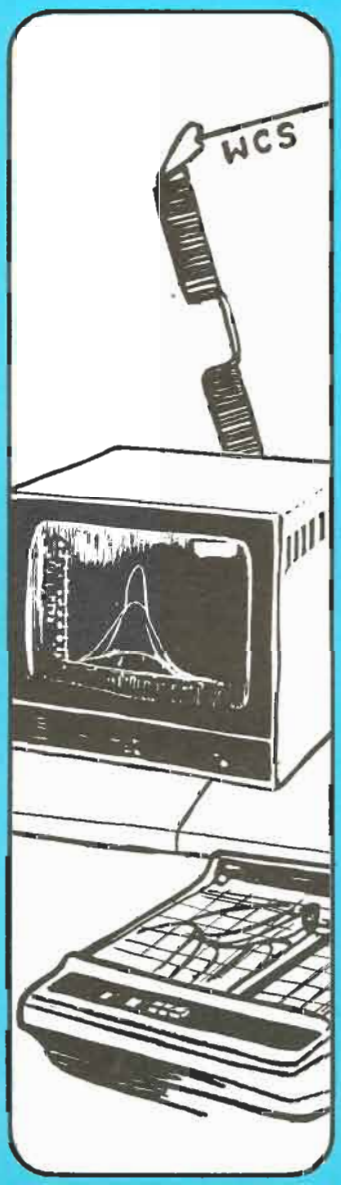
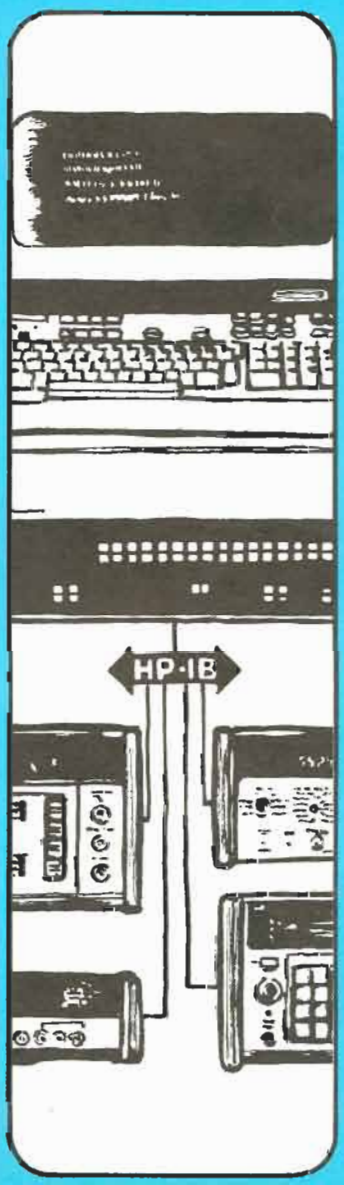
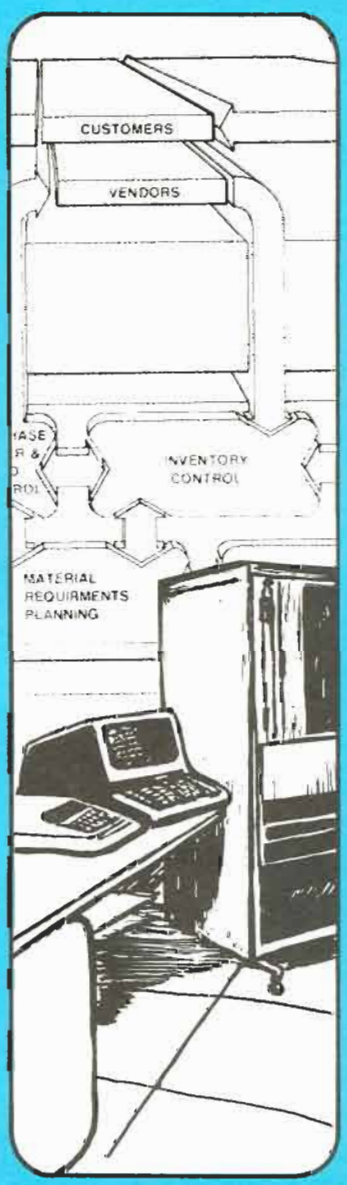
HEWLETT hp PACKARD

## Computer Systems

# COMMUNICATOR




issue no. 14

# HP Computer Museum
# www.hpmuseum.net

**For research and education purposes only.**

# EDITOR'S NOTE

Here's another great issue of the COMMUNICATOR 1000 to help you get the most from your HEWLETT-PACKARD COMPUTER SYSTEMS. It contains the first Communicator Index of all previous issues with a Cross Reference by subject category and issue. And better yet, the index pages are perforated so that you can remove them for convenient reference until the next index comes out.

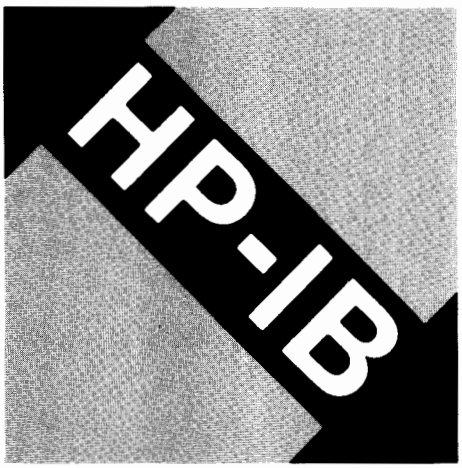
Other features in this issue talk about disc mapping, swapping tracks and segmentation in operating systems, and the new high reliability power supplies for 21MX series computers.

We at Hewlett-Packard are doing our best to keep you informed about the HP 1000. How about letting us know how well we measure up to your expectations? Drop a line to:

EDITOR

COMPUTER SYSTEMS/COMMUNICATOR 1000
HP Data Systems Division
11000 Wolfe Road
Cupertino, CA. 95014


# CONTENTS



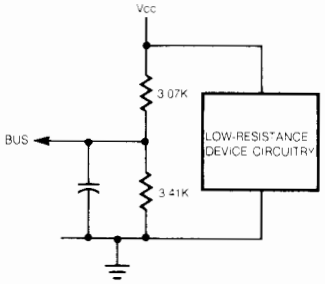



# HP-IB TREKIE ARTICLE #4

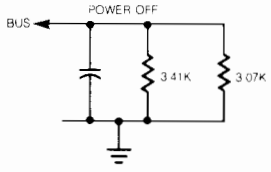
## POWERED-OFF DEVICES

*Larry W. Smith/DSD*

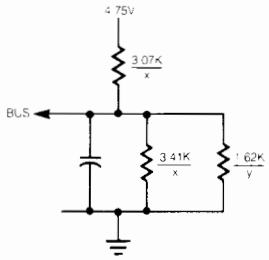
The standard (i.e. IEEE 488-1975) indicates that an HP-IB system will operate with only one more than half the devices powered on. This guideline is correct only if the open collector line can rise to a valid high threshold voltage when powered-off devices are loading the signal lines, suppose that a device's internal circuitry loads its power supply to the extent that its Vcc line is essentially grounded when the power is off. Then the worst-case resistive load on the bus could consist of the two termination resistors in parallel:




With this condition, a powered-off device could be equivalent to a 1.62K resistor.




The worst-case ratio of powered-on to powered-off devices can be found by requiring that the open-collector lines must rise to at least 2 volts to guarantee a high level. If X is the number of devices off and Y is the number of devices on, the equivalent resistive circuit is




solving for a bus voltage of at least 2 volts gives $X \geq 4Y$. At least four devices must be on for every one that is off. Note that this analysis assumes that there is no D.C. loading due to powered-off transistors, etc. which would be in the receiver circuit and connected directly to the bus line. Also note that although the bus will continue to work with one-fifth the devices powered off, the performance will be sharply reduced since the open collector rise times will be longer.

# HELPFUL HINTS ON USING QUERY

*The Editor*

In this article you'll find some helpful hints on using QUERY. We are very pleased to be able to pass information on to "you", our readers, when we have the opportunity. I'd like to thank ***Dave Welborn***, one of our customers from the south, for giving us this opportunity.

Hint 1: If your HP 2644/2645 terminal has the alternate character set defined, you may have a bit more security when using QUERY in the presence of unauthorized users. The security code, level word, and data base name can be somewhat protected by issuing a "CONTROL N" before each response. This turns on the alternate character set, yet still allows the proper commands to reach the CPU.

Hint 2: In QUERY report procedures intended for output on 264x terminals, enhancements can be used to accent totals:

```
T3,"sub total",53;
T3,"Esc&dJ",57;
T3,TAMT,70,E1,ADD;
T3,"Esc&d@",74;
```

Hint 3: Multiple versions of QUERY can be automatically created with the use of the transfer file you'll find listed below. This transfer file will create copies of QUERY that release the ID segment of the main program when completed. Before using this, the "SP" command must have been issued for QUERY, and the type 6 file renamed to QUE00. Also, the QUERY main program must be removed from the LOADR. This scheme will work when :RU,QUERY is issued from LU#1 to LU#99.

```
QUERY   T=00004 IS ON CR00002 USING 00002 BLKS R=0019

0001    :LOG,0G
0002    :SV,1
0003    :SV,QUE00,QUE00,,,SEL00
0004    :CR,3,0G,/,10
0005    :CA,4,-10,*,3G,+,0G,*,256
0006    :CA,-34:P,-34P,+,3G
0007    :CA,-33:P,-33P,+,4G
0008    :RN,2G,1G
0009    :RP,1G
0010    :RN,1G,2G
0011    :CA,-18:P,-18P,+,3G
0012    :CA,-17,:P,-17P,+,4G
0013    :CR,5G::2:2:100:128
0014    :DP,YOU ARE USING QUERY VERSION:,1G,ON LU#,0G
0015    :DP,USE SELECT-FILE=,5G
0016    :RU,1G,0G,0G
0017    :DF,1G,8
0018    :PU,5G
0019    :SV,0
```

Have any helpful software hints you'd like to have passed on? If so, please contact me:

The EDITOR
COMMUNICATOR 1000
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

# KNOW YOUR RTE — PART 8

*Mr. RTE*

This is the 8th of a series of articles in the **COMMUNICATOR** dealing with the inner works of HP's RTE systems. These articles go into some detail on how the system works; therefore, you should have already read and become familiar with the material in the RTE reference manual to your system.

Due to popular request, in this issue we will discuss the dispatcher.

As you will remember from prior articles in this series the dispatcher or DISP is entered at entry point $XEO. We have already discussed the list processing in DISP associated with the list headed at $ZZZZ. Now we will discuss actual program dispatching.

In the general case the dispatcher insures that a program has all the resources it needs to execute and then passes control to it. The resources the dispatcher is directly involved in is main memory or a partition.

It is also involved in reentrant memory processing and swap track management. For this discussion we will now introduce the following terms:

Contender:
Program from the schedule list that is currently being considered by the dispatcher.

Current Executing Program:
Program that was using CPU cycles at the last non-privilege interrupt. This programs ID segment address is located in XEQT (1717 on system base page). If there is no such program then XEQT is zero and the system was in the idle loop.

Resident Program:
The program currently residing in the disc resident memory area (partition) being considered.

## SO ALRIGHT ALREADY — HOW DOES IT DO IT?

All programs that are ready to run with the possible exceptions of not being in main memory and of not having their reentrant memory restored are organized by priority in the schedule list (head at SKEDD 1711 on the base page).

The first thing the dispatcher does is to examine the entry point of $LIST (see KNOW YOUR RTE part 1). The dispatcher will examine the schedule list only if the status of some program has changed. Since, to change a program's status $LIST must be called, the dispatcher sets $LIST to zero after a dispatching decision has been made. It can then test $LIST for zero on entry to see if anything has happened that might change the decision. So, if $LIST is zero, the current executing program is dispatched.

If $LIST is not zero, something has happened since the last dispatch resolution and a new decision must be made based on the current status. In this case the head of the schedule list is examined and tested for:

a. zero — in which case the idle loop is set up and entered.

b. (RTE III only) it is a program being held for a segment load because there is no $XSIO call available to load the segment, in which case the next entry in the list is examined.

If the list entry cannot be dispatched for any reason the dispatcher checks the next entry in the same way. For this reason we will assume in the rest of this discussion, that only the program we are testing is in the scheduled list, and that it is not the first such entry.

Once a program becomes a contender the dispatcher sets up pointers to the items of interest in its ID segment (priority, type, etc.). If there is a current executing program and it is scheduled, the dispatcher compares its priority with the contender's. If the current executing program is equal to or greater in priority than the contender the current executing program is dispatched (see below DISPATCHED).

If there is no current executing program or the contender has priority, the contender is further processed by its type as follows:

TYPE 1,4 — Memory Resident

The memory resident map is set up as the current user map (RTE III) and the program is dispatched.

TYPE ASSIGNED — (RTE III)

If the program is assigned to a partition, then further dispatch is conditioned on the type of the partition, i.e., type -2 real time disc resident or type -3 background disc resident, as described below.

TYPE 2,3 — Real time disc resident, Background disc resident

The program may or may not be resident in memory and if not may force a swap or some other disposition of some other program. In RTE-III the first thing done at this point is to allocate a partition in which to run the program. This proceeds as follows for all disc resident programs:

## FIND A PARTITION

The find a partition subroutine works with the memory assignment table the organization of which is given in Table 1.

This table is set up at generation time and is modified as the system runs. Partitions are linked together in the order of their availability in 6 different lists as follows:

1. Real time free list

   All available real time partitions are in this list in order of increasing size.

2. Real time allocated dormant list

   This list contains all allocated real time partitions which contain programs that have gone dormant either saving resources or serially reusable. Partitions are linked in order of priority of the program and, within the same priority in order of increasing partition size.

3. Real time allocated list

   This list starts at the end of the real time allocated dormant list and contains all the allocated real time partitions which contain active programs. Partitions are linked by priority and size as in the real time allocated dormant list.

4. Background free list

   Same as 1 but for background partitions.

5. Background allocated dormant list

   Same as 2 but for background partitions.

6. Background allocated list

   Same as 3 but for background partitions.

The "find a partition" subroutine starts by checking to see if the contender remains in the same partition it was in last. The last partition number is in word 22 of its ID segment and partition residence information is in word 2 of the memory assignment table (MAT) for that partition. If the program is still resident, the partition has been found.

If the contender is not in this partition, the contender is not in memory and a partition must be allocated for it.

If the contender must run in a particular partition (i.e., the RP bit is set in ID segment word 22) then, either the partition is free or the partition is current resident. If the partition is free, it is unlinked from the free list, relinked in the allocated list, and the required partition is found. When relinked in the allocated list the MAT entry word 2 is set to zero to show that the contender is not in memory yet. If the partition has a current resident, the resident program's memory bounds are set up in preparation for a swap out and the partition is found.

If the contender is neither resident in a partition nor assigned to a fixed partition the search carries on with a search of the free list, its number is put in word 22 of the contender's ID background). If a free partition is found it is unlinked from the free list, its number is put in word 22 of the contenders ID segment and the partition is put in the allocated list with its word 2 set to zero to show that a load from disk is needed, and a partition is found.

If a free partition is not available the search continues with a check to see if the resident in the partition the contender was last loaded into has an RCF (read completion flag) equal to 3 (see Table 1). If so the resident program's memory bounds are set up for a possible swap and the partition is found.

At this time the allocated dormant list is searched, for a partition with one of the following conditions:

a. Not reserved.

b. It is large enough.

c. Its resident is not locked into memory (CL bit in its ID segment word 15 not set).

d. Its resident is either of lesser priority (higher numerical priority) or the resident must not be scheduled if the contender is of lower priority.

If all these conditions are met for some partition then its number is put in the contender's ID segment word 22. The residents memory bounds are set up for a possible swap and a partition is found.

If no partition is found the dispatcher indexes to the next program in the schedule list and tries to dispatch it.

Table 1. Memory Assignment Table (MAT) Entry

| WORD | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Linkage pointer to next entry | | | | | | | | | | | | | | | |
| 1 | Priority of current resident program | | | | | | | | | | | | | | | |
| 2 | Current resident's ID-segment address | | | | | | | | | | | | | | | |
| 3 | | | D | | | | Beginning page of partition | | | | | | | | | |
| 4 | R | | | | | | Number of pages in partition | | | | | | | | | |
| 5 | RT | | | | | | | | | | | | | RCF | | |

D = Resident is dormant — save resources completion
R = Partition is reserved
RT = REAL TIME PARTITION
RCF = Read completion flag
0 - Read in progress
1 - Program is resident
2 - SWAP out or segment load in progress
3 - SWAP out complete but program still resident

## AFTER A PARTITION IS FOUND

After a partition is found the dispatcher must determine if the allocated partition has a resident in it. If so and if the resident is the contender, the RCF (see Table 1) flag must be checked.

If the RCF = 3 then the contender is still in memory. In this case the RCF is set to 1, the contender's swap tracks are released and it is dispatched.

If the resident is not the contender then a check is made to see if resources are available to load or swap a program to or from this type of partition. If the resources are currently free or are busy with the current partition, then a call to the swap check subroutine is made. The resources required to load or swap a program are the I/O call and its control parameters. These are busy from the time a call is made until it either completes or is cancelled. The dispatcher has two sets of these resources, one for background and one for foreground.

The swap check subroutine performs checks on the resident to determine which of 4 exits to make. These exits are the following:

1. Clear or cancel the current load, that is, the wrong program is being loaded and the load should be cancelled.

2. Swap the resident out, it must be and can be saved.

3. Load the contender, the resident is overlayable.

4. Try the next entry in the MAT, this partition contains an unswappable program.

Swap check makes the "cancel load" return if the resident is I/O suspended and of lower priority than the contender, and the RCF flag = 0 (indicating it is being read in).

The swap return is made if no swap tracks are assigned (ID word 28 = 0), the point of suspension is not zero, and if one of the following occurs:

a. The resident does not have priority.

b. The resident has priority but is dormant and in the time list, and it is to run more than swap delay time from now.

c. The resident is I/O suspended with a buffer outside of its memory area.

The "point of suspension" check indicates that the program was executed since it was last loaded into memory.

The load return is made if either (a) or (b) above is true but either swap tracks are assigned or the point of suspension is zero (indicating the disc copy is still unchanged). The load return is also made if the RCF (see Table 1) is 3.

The swap check routine makes the "not usable" return (return 4) if:

a. The core lock (CL) bit is set in the resident's ID word 15.

b. The RCF = 2, indicating a swap or segment load in progress.

c. The resident has priority and is dormant in the time list to run in less than swap delay time.

d. The resident is scheduled and has priority.

e. The resident is I/O suspended for read in (RCF=0) and has priority.

f. The resident is I/O suspended and the buffer is within the area to be swapped.

## AFTER SWAP CHECK

If the swap check indicates a clear is required, RTIOC is to be called at $IOCL to clear the on-going disc access, but first the program is rescheduled and the partition is relinked into the free list for either background or foreground partitions as required, and the program resident flag is cleared in the MAT entry (word 2).

If a swap return is made, the internal routine PREST is called to allocate swap tracks and set up the I/O request. If this occurs without problems the program swap is started by calling RTIOC at $XSIO, the RCF is set to 2 and the schedule list scan is continued.

PREST sets up for a disc load or swap as follows:

1. Sets memory bounds for the program.
2. If swap, get swap tracks if required and sets SMAN in the ID-segment (word 28).
3. Sets the initial disc address.
4. Calls setup to build the triplet for the load.

PREST checks the following options:

1. Short ID-segment (BG-segment load).
2. The "All of Core" bit causes the whole area to be swapped along with all of the area base page.
3. If swap then the first word is always the area boundary.
4. If swap and no track assigned then swap tracks are allocated.

### ABNORMAL EXIT

A JMP is made to X0035 if no disc tracks are available for swapping. This continues the schedule list scan with the next program.

If a load return is made from swap check, a check is made to see if the load resources are available. If not and the partition is empty it is relinked in the free list. In any case the schedule list scan is continued with the next program.

If the "load resources" are available and the priority of the partition resident is to change, the partition is relinked in its proper location by priority. The system base page words RTDRA, AVMEM, BRDRA and BKLWA are set up for the contender. The contender is made the new resident and is I/O suspended.

PREST is called to set up the I/O parameters and $XSIO is called to start the load. After $XSIO returns the RCF is set to 0 and the schedule list is rescanned from the top.

On the "swap not possible" return, the "find partition" routine is re-entered in an attempt to find an available partition.

### READ/SWAP COMPLETION

When the requested I/O to load or swap a program has completed RTIOC returns control to the dispatcher at the location indicated when the $XSIO call was made. The dispatcher then saves the status of the I/O completion, steps the RCF flag (0 goes to 1 and 2 goes to 3, see Table 1), clears the "I/O resources in use" flag, and sets the "segment load wait" flag if background completion. If a swap out just completed $LIST is set non-zero to force a schedule list scan and control is passed to $XEQ to scan the schedule list. In the case of read completion the RCF is forced to 1, the program is rescheduled via a $LIST call and if no read errors were detected control passes to $XEQ.

If there were read errors the program is aborted via a call to $ABRT and control is passed to $XEQ.

### DISPATCHING A PROGRAM

Once a program is in memory and ready to be dispatched the dispatcher:

a. Sets up the user map.

b. Sets up the base page words at XEQT if required.

c. Sets up the required base page words RIDRA, AVMEM, BKDRA and BKLWA.

d. Checks the RM and RE bits in it's ID word 21 and if they are set calls $RSRE in the EXEC to restore moved memory to the reentrant TDB's. If the RE bit is set the memory protect fence is also lowered to below the resident library.

e. Sets the memory protect fence and enters the program via $IRT in RTIOC.

The above discussion centered on RTE-III and foreground programs. To generalize it to RTE-II we need only realize that the number of partitions is limited to one for any given program. Background programs are handled the same as foreground programs except that the default swap boundary is all of memory and even this is handled outside of the dispatcher by setting the AM bit in ID word 15.

This concludes our discussion of the dispatcher. Write and tell us if you want more and of what.

## RETURNING RTE-III MEMORY SIZE IN NUMBER OF PAGES

*Larry W. Smith/DSD*

Many of our RTE-III users make extensive use of the DMS system for purposes external to the operating system. The below assembly FORTRAN callable subroutine extracts from the memory allocation table ($MATA) the number of 1024-word pages of memory the operating system is running under and returns the integer result as a function:

```
            ASMB,R,L   ••• RETRIEVE RTE-III MEMORY SIZE •••
00000               NAM MEMSZ,7

                    ENT MEMSZ

                    EXT .ENTR,$MATA



 DESCRIPTION
 -----------


  THIS INTEGER CALLABLE FUNCTION SUBROUTINE ALLOWS THE RTE-III
 TO DETERMINE HOW MANY 1024-WORD PAGES OF MEMORY THE OPERATING
:SYSTEM IS USING.

:CALLING SEQUENCE
 ----------------

  ISIZE = MEMSZ(:DUMMY)

    ISIZE ---> RETURNED NUMBER OF PAGES OF MEMORY

    DUMMY ---> ANY DUMMY VARIABLE OR CONSTAHT


00000 000000  ISIZE NOP             STRICTLY A DUMMY PARAMETER.

00001 000000  MEMSZ NOP             < ENTRY & EXIT POINT >
00002 016001X       JSB .ENTR      RETRIEVE CALLERS ADDRESSES.
00003 000000R       DEF ISIZE
                    .
00004 066002X       LDB $MATA      GET HEAD+1 OF MEMORY ALLOCATION TABLE.
00005 046033R       ADB =D-1       GET NUMBER OF 6-WORD ENTRIES.
00006 160001        LDA <,I        GET #ENTRIES.
00007 003004        CMA,INA        NEGATE FOR COUNTDOWN.
00010 072034R       STA ENTRY      SAVE FOR COUNTDOWN.
00011 006004        INB
00012 046036R NEXT  ADB =D6        SET TO ADDRESS OF FIRST POSSIBLE ENTRY.
00013 160001        LDA 1,I        TRICKLE DOWN THE TABLE
00014 052035R       CPA =D-1        TO FIND THE END (-1)
00015 026022R       JMP END          OF THE FIRST UNDEFINED PARTITION.
00016 036034R       ISZ ENTRY      LAST ENTRY?
00017 002001        RSS            NO, TRY NEXT ONE.
00020 026022R       JMP END        LAST ENTRY.
00021 026012R       JMP NEXT       TRY NEXT ENTRY.

00022 046037R END   ADB =D-3       ADJUST TO GET ADDRESS OF START PAGE#
00023 160001        LDA 1,I        GET PAGE# WITH SOME OTHER JUNK.
00024 012040R       AND =B1777     ISOLATE TO PAGE#
00025 072000R       STA ISIZE       AND SAVE.
00026 006004        INB            POINT TO ADDRESS OF PARTITION SIZE.
00027 160001        LDA 1,I        GET PARTITION SIZE & OTHER JUNK.
00030 012040R       AND =B1777     ISOLATE TO PARTITION SIZE.
00031 042000R       ADA ISIZE      ADD IN START PAGE NUMBER
00032 002004        INA             AND ADJUST RELLTIVE TO 1.

00033 126001R       JMP MEMSZ,I    RETURN TO CALLER.

00034 000000  ENTRY NOP            CURRENT ENTRY?

...LITERALS...

00035 177777
00036 000006
00037 177775
00040 001777
                      END
NO ERRORS •TOTAL ••RTE ASMB 760924••
```

If the memory management scheme is dependent upon the total amount or the last physical page of memory, this subroutine could be used to quickly determine this. This subroutine is also used in the LOCUS Program 'LUPRN' which lists any RTE device configuration which has recently been updated and modified to include new devices and printing of the time-base select code, privileged interrupt select code, and memory size.

## SWAPPING: HOW IT'S DONE, HOW LONG IT TAKES, WHEN TO WORRY ABOUT IT, WHAT YOU CAN DO

*Lyle Weiman/DSD*

This article will discuss how you can control the program swapping that occurs due to the contention for system resources, and how to determine the amount. In a multiprogrammed environment, programs must compete with each other for system resources (CPU time, partition space, buffer memory, disc space, etc.) If a program can't have a resource (for whatever reason), it is suspended until that resource is available; if that program is disc-resident, then its partition may be useable by another program. If so, then the first program's partition of memory is copied to a disc track ("rolled out") and the other program is brought in from the disc. The whole operation is called a "swap". Obviously, when the resource the first program wanted becomes available, the whole process *may* be reversed. The specific factors which control whether this happens are described in the article. A "swap" can take as much as half a second, during which time that partition is idle, and the DCPC cycles slow down the CPU. Minimizing needless swapping is therefore important to you.

Since a "swap" is only necessary when the number of disc partitions is less than the number of running programs and possible only when at least one partition is large enough, and its occupant is not "locked in" for any reason, the simplest thing you can do to decrease swapping is to increase the number of available partitions. In RTE III, this is usually possible by adding physical memory, but in either RTE it can sometimes be done by generating a smaller RTE

(fewer ID segments, EQTs, smaller resident library, etc., but it is seldom useful to decrease SAM; this can often increase swapping, as you'll see later) and defining smaller partitions.

In RTE II, this also means balancing the swapping between real-time and background programs (more on how to do this later). Insofar as possible, try to keep the smaller programs in the real-time area, large programs in background: small programs can swap very quickly.

### Program Priority

This is a very "loose" form of control over swapping. A program can almost always be swapped when a higher-priority program becomes "ready" (state 1). Exceptions occur with any one of the following conditions:

1. There are insufficient contiguous disc tracks available on either the system or auxiliary disc to swap out the current partition resident. When this is the case, your RTE will appear to be partially asleep. You can obtain system attention and most commands will work, but no disc-resident programs can run until the current partition residents either terminate or are aborted. Since $$CMD is a system program which handles LU, DN, UP and IO commands, you may be unable to use them if this program is disc-resident. If you check program status, you'll find that all the programs you called for are scheduled (state 1); notice that this blockage can result without any program explicitly being in state 5 (unavailable disc suspend).

   You can relieve the situation by releasing disc tracks from the following:

   a. A program which obtains them without releasing them (e.g., EDITR). The program must be dormant.

   b. The LG area. (LG, 0). This will only help if there are LG tracks allocated.

   c. The current resident of one of your partitions by aborting it. The partitions must be background, in order for EDITR, FMGR, FTN4.

or,

2. The current partition resident has locked itself in memory with the EXEC memory-lock call or the partition is "assigned" to this program and not the "newcomer".

or,

3. The resident is suspended for regular I/O. Note that the other forms of I/O do not cause this (buffered output, class — or re-entrant I/O).

When a higher-priority program becomes "ready" (scheduled) then if a swap is needed to give that program a partition in which to run, the lowest-priority program which is in a partition large enough for the "newcomer" will be swapped out. This tends to force most of the swapping to be done on the lower-priority program, allowing the others to continue.

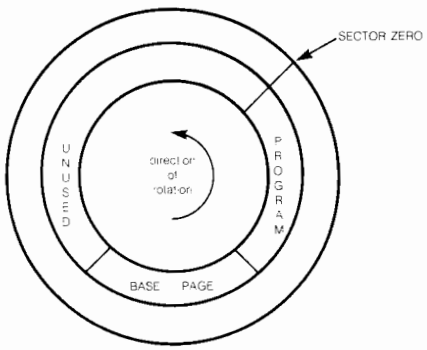
Program priority does not *prevent* low-priority programs from swapping high-priority ones. For example, if the high-priority program must wait for some resource (class I/O completion, buffer memory, I/O completion, etc.) then it may be swapped by *any* program in the "ready" (scheduled) state which needs the partition.

If, for example, there is one I/O-bound program (assumed to output only to buffered devices), a CPU-bound program having a lower priority, and one partition, then the I/O-bound program is placed in state 3 (general wait) each time its output data exceeds the high buffer limit. Then it is swapped out by the lower priority program. The lower priority program runs until the I/O device empties enough data blocks so that the total amount of SAM is less than the lower limit where the reverse swap occurs.

You can see how choice of the buffer limits can affect swapping. The high limit should be high enough to allow several records to be buffered, so that a program can perform useful work while it "owns" a partition, yet not so close to the total available SAM size, or your system may quietly "go to sleep" if the buffered device should run out of tape or paper (thus "clogging" SAM). The low limit should be sufficiently lower than the high limit to prevent frequent reverse-swaps, without allowing pauses to occur in the output; i.e., large enough so that when reserve-swaps do occur, the device still has data ready to output.

### How A "SWAP" is Done

When the criteria for a swap are all met, then the partition resident is swapped out, base page area first, then program area, like this:



Notice that there are unused sectors following the base page area; this is true if the program size (rounded at the next even sector) plus base page (similarly rounded) add up to less than the number of sectors per track. During the time these sectors are passing under the disc head, the software is setting up DCPC to transfer the program area. Depending on CPU operating system and disc being used, the minimum time the software requires is about that for 8 sectors to pass. If the program is small, say, 5K or so, then it can be swapped out in an average of 1-1/2 revolutions using a 7905 or 7920. For the 7900, transfer of one whole track requires two revolutions, so programs of slightly more than 2K can be rolled out in 1-1/2 revolutions, (37.5 ms), 5K programs require 2-1/2 revolutions. An extra revolution is required if your program is too large to fit on a single track (including base page). The above discussion also applies to the incoming program of the swap if it resides on a swap track. Programs included at RTE generation time, or saved by FMGR SP command, are packed so that several programs can be stored on the same track, but the optimization may still hold if the base page part comes around, and if both parts are on the same track.

## IT PAYS TO KEEP PROGRAMS SMALL

Programs larger than 6K require several revolutions to transfer in or out of memory. Up to a track length, you don't gain much by shortening your programs, but at a track length each transfer incurs an extra "seek"; this takes at least one extra revolution.

We don't suggest you spend a great deal of time squeezing every last word of code out of every program. You can, however, realize great savings by telling the operating system that your program does not use "undeclared memory", i.e., the area of memory between the last word the relocating loader saw and the end of the partition. You do this by including the EXEC (22,2) call somewhere in the initialization code.

It's not nice to lie to your operating system: if you tell it you don't use undeclared memory, but you really do, you'll be sorry. The first time your program is swapped by a larger program, you'll lose your data area, and it will "blow up". If you don't know whether a program uses this area or not, add the EXEC (22,2) call, and test it by forcing swaps by larger programs.

In RTE III, the best you can do with this method is save a little less than a page; this is because you must tell the LOADR specifically if more pages are required. In RTE II, you can save several pages, and the method is most useful if you have a large background area but run small programs in it.

You may choose to reduce swapping by more drastic means such as locking programs in memory with CALL EXEC (22,1). Before doing this, you should be sure that your critical-response-time programs (if any) will still have a partition to run in.

Another way to control swapping is via the "swap delay". This parameter, which is set at generation time, is only useful when you have programs which run on a time-scheduled basis, or offset-schedule themselves at times waiting for something. If the following are all true:

- the current resident of a partition is waiting for a particular time of day, and
- the program which is to be run in that partition has a lower priority than its current resident, and
- the time remaining is less than or equal to the current swap delay

then the current resident remains in the partition (is not swapped out); that partition remains unchanged and unused until the time arrives for its resident to run.

You adjust the swap delay so that it is longer than the maximum time taken to swap a program out and roll another one in. If it is too short, then, the system will waste time swapping the current resident out. If it's too long, then the partition will stand "idle" longer than necessary. Of course, when it's time to run that program, the system is smart enough to cancel the "swap" if the contents of the partition have not been changed by the roll-in of the other program. The other program can be run when its time comes up; otherwise, the roll-in is cancelled and the original program is brought back in again.

Besides the base-page-first transfer technique, the system attempts to minimize head movement by allocating all "system" tracks (used for LG and swapping tracks) beginning at the last available tracks. If you have a single-drive system, this tends to keep the disc head at high-numbered tracks for FMP reads, directory track access and swapping, with occasional excursions to low-numbered tracks when a program included at generation time is called for.

If you have two disc drives, you should use one for the system disc and one for the auxiliary disc. Since the drives are separate, swapping can occur on one while program-

loads can occur on the other without incurring a lot of "seek" time overhead. Minimizing head movement dramatically reduces swapping overhead.

If you still want more speed, consider using two disc controllers. This avoids the bottle-neck of the single disc driver which can only process one disc request at a time (multiple EQTs are not the answer). On rare occasions, you may see simultaneous data transfer on both discs; this may cause data overrun on a 21MX computer; the driver automatically retries, but this could slow you down. The 21MX-E can handle both DCPC channels at 7905 data rates. You will have to add the track map for one of the discs yourself: the generator builds one for the system disc. See the RTE II or III reference manuals, Appendix B, for a description of the track-map table. If the discs are different (e.g., 7900, 7905), you will need to generate your system with both drivers.

If both discs are the same (or if they use the same driver, such as 7905 and 7920) you must make a second copy of the driver, renaming it and giving it different entry points (the number part should agree with the number part of the track map, e.g., $TB31 goes with DVR31 which has entry points I.31 and C.31, $TB32 goes with DVR32 which has entry points I.32 and C.32). This is because the disc drivers cannot handle more than one controller.

## Determining How Much Is Too Much

The discussion so far is useless unless you have some way of evaluating the merits of a particular choosing of the parameters. You may have your own methods which are application-dependent. For those of you who only know that it seems to work, but you'd like to know how much more it can handle, read on.

First, you must realize that "too much" is a subjective judgement. You can make rough measurements which can indicate whether your system is spending a great deal of time swapping, but no quantitative measurements are possible (unless you modify the system to keep statistics on its swapping).

In a previous issue of the *Communicator,* a CPU utilization program listing was shown. You should modify that program a little for these measurements by removing the READ/WRITE calls, thereby shortening the program. See the RTE II or III reference manuals for how to use CNUMD to convert internal binary to ASCII. There is one floating-point number (calculated CPU utilization) which you will have to print in two parts:

```
IPRCT  = IFIX(CPULOD)
KPERCT = IFIX((CPULOD-FLOAT(IPRCT))*100.0+0.5)
```

Convert IPRCT and KPERCT into ASCII; print IPRCT first, then a decimal point then only the last two digits of KPERCT.

The key to these measurements is to realize that, if your system is spending a great deal of time swapping, then the CPU will be idle quite a bit, but a low-priority disc-resident program will not get much time to run.

First, modify your measurement program to lock itself in memory as soon as it is run; of course, its priority should be 32767. If possible, assign it to a small, seldom-used partition. With your applications programs running at typical tasks, run your measurement program several times, and take an average. The CPU utilization printed represents the percentage of time the computer is doing useful work (the rest of the time, it would be in the idle loop if your measurement program wasn't there).

Now, remove the memory-lock call and load your measurement program again, but do not assign it to a partition. When it is run, it must compete with other programs for partition space (it does not require any other resource; the CPU time it uses would otherwise be spent idle, due to its very low priority). With your applications programs active, run your measurement program again several times, again taking the average. Compare the two results: if they are very close, the system is not swapping very much. The farther apart, the more time your system is spending doing swaps.

You can repeat these measurements, varying the measurement program's priority and making it real-time and background. The difference between measurements at priority levels can be used to determine which groups of programs are "hogging" the CPU and swapping capacity of the system; the difference between measurements taken as a "real time" vs. a background program will tell you if your real-time partitions are being swapped as much as your background.

You can adjust the other parameters mentioned in this article, re-run the tests and determine optimal values. Be careful about parameters which are picked up during boot-up such

as the swap delay. The swap delay is stored in the high 8 bits of base page location 1736, in units of centiseconds: these will have to be modified on disc and then the system re-booted.

You may want to use the on-line partition redefinition program (DFINE) in the contributed library to investigate the best use of memory for your system (RTE III only).

## SEGMENTATION

*Lyle Weiman/DSD*

Segmenting your programs provides an inexpensive means of attaining some of the advantages of virtual memory without the hardware cost. That is, you can write programs which are larger than the size of your background partition provided that all of the program does not need to be in memory at once. It is the programmer's responsibility to divide the program into portions that need to be in memory at the same time, and to provide EXEC calls to bring each portion into memory when required. This article will describe how.

First, the programmer (mentally) separates parts of his code into functional units which are only needed at certain times. Data areas common to these units are also separated. A typical functional unit breakdown may look like the following:

Resident Main —
Contains common data areas, RMPAR call to get parameters; loads initialization segment.

Initialization segment —
Sets defaults for parameters, checks parameters for legality; may open or create files. Loads first processing segment.

First processing segment —
Reads first input record. An interactive program would ask the operator for the next command. After inputting the first data record or command, it may determine the record or command type. Determines the segment to load according to record or command type.

Other processing segments —
Process record or command, provide full error checking; full processing may require other segments to be loaded. When processing is complete, the first processing segment is loaded (unless an "END" statement has been read). The "END" or final statement may require that another segment be loaded, which closes files and does other "cleanup" processing. This segment contains the CALL EXEC (6) which terminates the program.

The advantages of segmentation should be apparent from this example:

The subroutines CREAT, OPEN and CLOSE are in separate segments. Initialization code does not "take away" available memory space from the other main processing code.

A program can have many functional capabilities and yet still run in a small partition, since the code for each separate function can exist in a separate segment.

Additional functional capability can be added simply by adding another segment, and modifying the first processing segment to load that segment when required.

Although some Assembly code is required (this will be shown later) the processing code may be written in FTN-IV. FTN-II, ALGOL, or Assembly languages. Typically, the process of segmenting a high-level language program is merely one of identifying the functional relationships and common data areas to the main. Assembly language is used to provide the linkage between data areas in the resident main and the various segments.

Notice that the data areas must be kept in the resident main. This is because, when a segment is loaded, the segment currently resident is *overlaid*: it is not swapped out, and any data kept in the segment is *lost*. Since certain data areas are only needed by the segment while it is processing the command or input record, the following guidelines are offered for deciding what data to keep in the resident main:

- if the data is used by more than one segment

or

- if the segment will need the data again later the next time it is loaded (remember, loading a segment always brings in a "fresh" copy)

  or

- if you're in doubt about whether either of the above conditions are true, consider the following:

  The memory size your segmented program requires is the sum of the resident main and *largest* segment size. If a particular data area is used by the largest segment, then it might as well be kept in the resident main, and you may save yourself the trouble of determining whether the criteria above are met.

Figure 1 shows how memory looks at various times in the execution of a segmented program. Figure 1(a) shows only the resident main: no segment has been loaded yet. Figure 1(b) shows how the first segment looks ("first" simply means the first segment which the resident asks EXEC to load; "second" means the next segment loaded, etc. "First" is not necessarily named SEG1, DSG1, etc.) when loaded into memory.

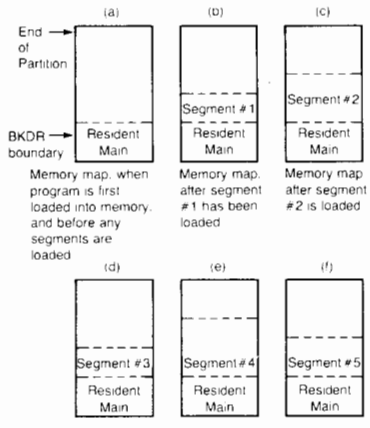



Figure 1. Memory Maps for Various Segments

## RESIDENT AND SEGMENT MAIN FORMATS

The resident main *must* be a type 3, 11, 19 or 27 (background disc — resident) and you *must* include in the END statement (for Assembly programs) the label of the first instruction to be executed when the program is run. For example:

```
      ASMB,L
            NAM RMAIN,3,90
            EXT EXEC
START       LDA  B,I                    get first parameter
             .
             .
             .
            JSB  EXEC                   load first segment
            DEF  *+3
            DEF  D8
            DEF  SEG1
            D8   DEC8
            SEG1 ASC3,SEGM1
            END  START
```

Your resident main may be a FORTRAN program, e.g., whereby you don't need to specify the 1st statement to be executed in the END statement.

```
      FTN,L
            PROGRAM RMAIN      (3,90)
            INTEGER  SEG1 (3)
            DATA     SEG1/2HSE,2HG1,2H  /

            CALL EXEC (3,SEG1)
            END
            END$
```

In the FORTRAN case, all data areas used by more than one segment must be specified in an assembly module. You force this module to be included in the resident main by including a dummy subroutine in it, which the FORTRAN main program calls. This subroutine may be a dummy, or it may load the first segment.

The segment mains have a similar format:

```
            NAM SEG1,5
            EXT (list of data elements)
START       EQU*
             .
             .
             .
            END  START
```

The essential elements here are that the module must be type 5, and that the END statement must include the label of the first instruction. The segment main may be FORTRAN, if you prefer, but then you'll have to solve the problem of linking to the resident main data areas, such as through COMMON. A much simpler solution is shown in the program example: the segment mains are all in assembly and they contain EXT statements these referencing these data areas.

They then call the high-level language (ALGOL, or FORTRAN) subroutines passing the data area addresses to the subroutine. All of the computation is done in a high-level language; the assembly modules merely serve to provide data- and control-passing between resident main and each segment. In order for the RTE generators to load the segments with the proper main, be sure to include at least one external reference in the segment to a symbol in the main.

## SUBROUTINES

A subroutine which is used in the largest segment could be located in the main, in order to reduce the size of that segment and hence reduce the overlay time (no memory space is saved, however). Other subroutines are best located in the segments where called.

To "force" a subroutine to be loaded with the main, simply put an external reference in the resident main; for example;

```
     NAM  RMAIN,3
     EXT  SUB1
RMAIN  EQU*
        .
        .
        .
       END  RMAIN
```

will cause "SUB1" to be relocated with the resident main. This is true of the on-line and off-line RTE generators and all relocating loaders.

## SYMBOL TABLES

Some programs have requirements for variable-sized data tables, such as the symbol tables in the compilers, assembler and loader. As various segments are loaded into memory, notice how the lower bound of available memory changes (see Figure 1). If your program requires symbol table area, you'll need to determine the largest segment, and start your table after its last word. The example shown at the end of this article shows how to scan the ID segments and determine the largest segment. In order to determine the address of the first available location, use the ID segment of the largest segment (the HIMAN word actually contains the last address +1 for this segment.) Since some segments may have "short" (9 word) ID segments, you should use the COR.A subroutine, described in the RTE-II and -III Reference manuals.

If you know the name of the largest segment, then you can scan the keyword block, looking for a match to the name. Simply call COR.A with the ID segment address when you find a match. If you don't know the name of the largest segment (for example, during development the sizes of the segments will change), then keep a list of the segment names, and scan the ID segments as described above, calling COR.A for each match and keep the maximum.

The example program shown, illustrates these points. There is an initialization segment, which determines the size of available memory, computes the number of entries in a symbol table, prints the table size and starting address, and loads the first processing segment. The first processing segment accesses two data items in the resident main: the first 14 characters of a message, and the terminal logical unit. It prints this part, and loads the second processing segment. The second processor prints the other half of the message and terminates. Note that the first half of the message terminates with a back-arrow (←), so that when run from a console the two parts of the message appear to be printed as a single line; when printed on a lineprinter (as shown here) they print as separate messages.

The initialization segment does one more thing: it adjusts the descriptor table for the two dimensional ALGOL array, changing the effective declaration to be as large as the partition can allow.

## PERFORMANCE CONSIDERATIONS

Each time a segment is loaded, one or more disc-read operations must take place. The amount of time required depends on segment size and the current location of the disc arm. Assuming no other system activity, the following represents typical transfer times:

| Segment size (octal) | Transfer time (ms) |
|---|---|
| <6000 | 50 |
| 10000 | 75 |
| 12000 | 125 |
| 26000 | 150 |
| 30000 | 200 |

These times were measured on a 21MX computer, using 7900 disc and RTE-III. Although the E-series using 7905 would be faster, the point here is that best performance is obtained when the fewest possible segment loads are required. You must balance memory size versus number of segment loads to gain overall performance. Usually, this is easy: You are given a fixed partition size, and you must fit your program into it.

## USING THE ON-LINE LOADER FOR LOADING SEGMENTED PROGRAMS

The fastest and surest method is to make a disc file containing FMGR commands, such as the following:

```
:OF,  <resident main>
:OF,  <1st segment>
:OF,  <2nd segment<
:OF   <last segment>
:LG,5                                  Note: Some segmented programs require many more LG tracks
:MR,% <resident main>                  move relocatable file for resident main to LG.
:MR,% <main subroutines>               move relocatable for subroutines to be loaded with resident main.
:MR,% <1st segment main>               move relocatable file for 1st segment main.
:MR,% <1st segment subroutines>        move subroutines required by 1st segment main

:MR,%  <2nd segment main>              move 2nd segment main
    .
    .
    .
:RU,LOADR,99,6,P3,P4,P5                P4 must end in 1. You may wish to specify extra memory and/or SSGA.
                                       See the RTE-II or -III reference manual for details on P3, P4, P5.
```

## MULTI-TERMINAL OPERATION

You may wish your segmented application program to be usable by several operators at the same time. The techniques used to provide multiple copies of the File Manager may be used:

```
:SP,FMGR                ←This only needs to be done once, after generation

:RN,FMGR,FM07.  }
:RP,FM07.       }
:RN,FM07.,FM20  }       Put this in your WELCOM file
:RP,FM20        }
:RN,FM20.,FM21. }
      .
      .
      .
:RN,FM25.,FMGR
```

Here, the terminal logical units are 7, and 20 through 25. There is nothing special about this naming convention, it's just easy to remember and use.

Notice that, although FMGR has nine segments, only the resident main needs to be copied.

## SEGMENTATION EXAMPLE—MAIN PROGRAM

```
0001                        ASMB,R,L
0002  00000                   NAM RMAIN,3 SEGMENTATION EXAMPLE,RESIDENT MAIN
0003                          SUP
0004                          ENT MSG1,MSG2
0005                          ENT [SYMT,[MAXS
0006                          ENT LU1
0007                          EXT EXEC
0008  00000             RMAIN EQU *
0009  00000 160001            LDA B,I         GET INPUT LU
0010  00001 002003            SZA,RSS         DEFAULT?
0011  00002 002404            CLA,INA         YES, USE DEFAULT
0012  00003 072011R           STA LU1
0013  00004 016001X           JSB EXEC        LOAD FIRST SEGMENT
0014  00005 000010R           DEF *+3
0015  00006 000010R           DEF D8
0016  00007 000030R           DEF SEG1
```

```
0017*
0018*      DATA AREA
0019*
0020   00010 000010   D8     DEC 8
0021   00000          A      EQU 0
0022   00001          B      EQU 1
0023   00011 000000   LU1    NOP
0024   00012 046505   MSG1   ASC 7,MESSAGE FROM,
0025   00021 052127   MSG2   ASC 7,TWO SEGMENTS.
0026   00030 046520   SEG1   ASC 3,MPASI
0027*
0028*
0029*
0030********************  STORAGE   FOR   ALL   SEGMENTS   **********************
0031*
0032*      THE CODE BELOW SIMULATES AN ALGOL ARRAY DECLARATION.  IT
0033*      IS SHOWN IN THIS EXAMPLE TO DEMONSTRATE HOW AN ALGOL
0034*      PROGRAM CAN BE GIVEN AN ARRAY OF INDEFINITE SIZE.  FOR EXAMPLE,
0035*      IN SOME SYSTEMS, THIS PROGRAM MAY BE RUN IN A VERY LARGE
0036*      PARTITION,  IN WHICH CASE IT WILL HAVE A VERY LARGE ARRAY
0037*      SIZE.  IN OTHERS, IT WILL RUN IN A SMALL PARTITION AND
0038*      HENCE HAVE A SMALL SIZE.  THE ACTUAL CODE TO CALCULATE
0039*      THE NUMBER OF ENTRIES IN THIS TABLE, AND THE TABLE
0040*      ADDRESS IS CONTAINED IN THE INITIALIZATION SEGMENT.
0041*
0042   00012          ENTSZ EQU 10          SYMBOL TABLE ENTRY SIZE.(I.E., 10 WORDS

0044*                     DUMMY ALGOL ARRAY DECLARATION
0045   00033 177776   [SYMT DEC -2          NEG. # DIMENSIONS (2-D ARRAY)
0046   00034 000000   [MAXS NOP             SIZE OF FIRST DIMENSION.
0047*                     NOTE:THIS SIZE IS COMPUTED IN
0048*                     INITIALIZATION SEGMENT.
0049   00035 177777         DEC -1          NEGATIVE OF LOWER BOUND
0050   00036 000012         ABS ENTSZ       UPPER BOUND,2ND DIMENSION
0051   00037 177777         DEC -1          NEG. OF LOWER BOUND, 2ND DIMENSION
0052   00040 000000         NOP             ADDRESS OF TABLE,CALCULATED IN SEGMENT
0053*
0054                        END RMAIN
**  NO ERRORS *TOTAL **RTE ASMB 760924**


0001                ASMB,R,L
0002   00000                NAM SEGLD,7 RTE-II/III  INTERFACE 4/12/76
0003*      EX5
0004                        ENT SEGLD,LIMEM
0005                        EXT .ENTR,EXEC
0006                        EXT COR.A
0007                        EXT LU1

0009   00000 000000   SNAME NOP             SEGMENT NAME ADDRESS
0010   00001 000000   SEGLD NOP             SUBROUTINE ENTRY POINT
0011   00002 016001X        JSB .ENTR       GET PARAMETER ADDRESSES
0012   00003 000000R        DEF SNAME
0013   00004 016002X        JSB EXEC
0014   00005 000010R        DEF *+3
0015   00006 000014R        DEF SLCOD
```

```
0016  00007 100000R        DEF SNAME,I
0017  00010 026034R        JMP SEGER      SEGMENT NOT HERE.
0018  00011 126001R        JMP SEGLD,I    RETURN TO CALLER
0019  00012 000006  D6     DEC 6
0020  00013 000002  D2     DEC 2
0021  00014 100010  SLCOD  OCT 100010     SEGMENT LOAD CODE, NO-ABORT BIT SET

0023  00015 000016  MSG    DEC 14
0024  00016 051115         ASC 14,RMAIN:ALL SEGMENTS NOT FOUND
      00017 040511
      00020 047072
      00021 040514
      00022 046040
      00023 051505
      00024 043515
      00025 042516
      00026 052123
      00027 020116
      00030 047524
      00031 020106
      00032 047525
      00033 047104

0026  00034          SEGER  EQU *
0027  00034 016002X         JSB EXEC       PRINT ERROR MESSAGE
0028  00035 000042R         DEF *+5
0029  00036 000013R         DEF D2
0030  00037 000004X         DEF LU1
0031  00040 000016R         DEF MSG+1
0032  00041 000015R         DEF MSG
0033  00042 016002X         JSB EXEC
0034  00043 000045R         DEF *+2
0035  00044 000012R         DEF D6

0037  00045 046520  SEG2   ASC 2,MPAS     FIRST FOUR CHARACTERS OF SEGMENTS
      00046 040523
0038  00047 000000  WHICH  NOP
0039  00050 000000  FWAM   NOP            FIRST WORD OF AVAILABLE MEMORY
0040  00051 000000  NWORD  NOP            # OF AVAILABLE WORDS.
0041  00052 000000  LIMEM  NOP            SUBROUTINE ENTRY POINT
0042  00053 016001X         JSB .ENTR
0043  00054 000047R         DEF WHICH
0044  00055 061657          LDA KEYWD      GET ADDRESS OF KEYWORD BLOCK
0045  00056 072160R         STA PNTR       STORE FOR LOOP
0046*
0047*      THIS LOOP SEARCHES FOR THE SEGMENTS
0048*      FOR PASSES 1 AND 2, AND OBTAINS THE FIRST WORD OF
0049*      AVAILABLE MEMORY FOR EACH.
0050*
0051*
0052  00057          LOOP   EQU *
0053  00057 166160R         LDB PNTR,I     GET NEXT ID SEGMENT
0054  00060 006003          SZB,RSS        LAST ONE?
0055  00061 026034R         JMP SEGER      ALL SEGMENTS NOT FOUND.
0056  00062 046164R         ADB =D12       ADVANCE TO NAME
0057  00063 016153R         JSB GNAME      GET FIRST TWO CHARS OF NAME
```

```
0058  00064 052045R        CPA SEG2       MATCH FIRST TWO SEGMENT NAME CHARACTERS?
0059  00065 002001  RSS
0060  00066 026104R        JMP LEND
0061  00067 016153R        JSB GNAME      GET CHARACTERS 3 & 4
0062  00070 052046R        CPA SEG2+1     MATCH 2ND PAIR OF SEGMENT NAME CHARACTER
0063  00071 002001         RSS
0064  00072 026104R        JMP LEND       NO, GO GET NEXT ID SEGMENT
0065  00073 160001         LDA B,I
0066  00074 001727         ALF,ALF
0067  00075 012165R        AND =B177
0068  00076 052166R        CPA =B61       PASS 1?
0069  00077 026112R        JMP PASS1
0070  00100 052167R        CPA =B62       PASS 2?
0071  00101 026106R        JMP PASS2
0072  00102 052170R        CPA =B111      INITIALIZATION?
0073  00103 026116R        JMP PASSI
0074  00104          LEND  EQU *
0075  00104 036160R        IS7 PNTR
0076  00105 026057R        JMP LOOP
0077  00106          PASS2 EQU *
0078  00106 162160R        LDA PNTR,I       FIND SEGMENT # 2'S FWA
0079  00107 016003X        JSB COR.A
0080  00110 072162R        STA PAS2M
0081  00111 026121R        JMP PASSM
0082  00112          PASS1 EQU *
0083  00112 162160R        LDA PNTR,I       FIND SEGMENT # 1'S FWA
0084  00113 016003X        JSB COR.A
0085  00114 072161R        STA PAS1M
0086  00115 026121R        JMP PASSM      SEE IF ALL SEGMENTS FOUND.
0087  00116          PASSI EQU *          INITIALIZATION SEGMENT
0088  00116 162160R        LDA PNTR,I     FIND INITIALIZATION SEGMENT'S FWA
0089  00117 016003X        JSB COR.A
0090  00120 072163R        STA PASIM
0091  00121          PASSM EQU *
0092  00121 062161R        LDA PAS1M      ALREADY HAVE SEGMENT # 1?
0093  00122 002003         SZA,RSS
0094  00123 026104A        JMP LEND       NO, CONTINUE
0095  00124 062162R        LDA PAS2M      ALREADY HAVE SEGMENT # 2?
0096  00125 002003         SZA,RSS
0097  00126 026104R        JMP LEND
0098  00127 062163R        LDA PASIM      ALREADY HAVE INITIALIZATION SEGMENT?
0099  00130 002003         SZA,RSS.
0100  00131 026104R        JMP LEND       NO, CONTINUE SEARCH
0101  00132          CLCLN EQU *
0102*
0103*      CALCULATE AVAILABLE SYMBOL TABLE AREA
0104*
0105  00132 062161R        LDA PAS1M      FIND LARGEST SEGMENT'S LAST WORD ADDRESS
0106  00133 064000         LDB A          GET LAST WORD OF SEGMENT # 1
0107  00134 003004         CMA,INA        SUBTRACT FROM
0108  00135 042162R        ADA PAS2M         LAST WORD OF SEGMENT # 2
0109  00136 002021         SSA,RSS        SEGMENT # 1 LARGER?
0110  00137 066162R        LDB PAS2M      NO, SEGMENT # 2 IS LARGEST.
0111  00140 062163R        LDA PASIM      GET INITIALIZATION SEGMENT'S LWAM
0112  00141 003004         CMA,INA
0113  00142 040001         ADA B
```

```
0114  00143 002020         SSA
0115  00144 066163R        LDB PASIM
0116  00145 176050R        STB FWAM,I     RETURN FIRST AVAILABLE ADDRESS HERE.
0117  00146 060001         LDA B          GET LAST WORD OF LARGEST SEGMENT
0118  00147 003004         CMA,INA        SUBTRACT FROM
0119  00150 041777         ADA LWAM        LAST BACKGROUND MEMORY ADDRESS
0120  00151 172051R        STA NWORD,I    RETURN # WORDS AVAILABLE TO CALLER
0121  00152 126052R        JMP LIMEN,I    RETURN TO CALLER.

0123  00153 000000  GNAME  NOP            SUBROUTINE TO GET NEXT NAME CHARACTERS
0124  00154 160001         LDA B,I
0125  00155 012171R        AND =B77577    MASK
0126  00156 006004         INB
0127  00157 126153R        JMP GNAME,I

0129  01777          LWAM   EQU 1777B      ADDRESS OF LAST WORD BACKGROUND MEMORY
0130  01657          KEYWD  EQU 1657B      ADDRESS OF KEYWORD BLOCK
0131  00000          A      EQU 0
0132  00001          B      EQU 1
0133  00160 000000   PNTR   NOP
0134  00161 000000   PAS1M  NOP
0135  00162 000000   PAS2M  NOP
0136  00163 000000   PASIM  NOP
      00164 000014
      00165 000177
      00166 000061
      00167 000062
      00170 000111
      00171 077577
0137                       END
**  NO ERRORS *TOTAL **RTE ASMB 760924**


0001                ASMB,R,L
0002*     FX4
0003  00000                NAM MPASI,5 SEGMENTATION EXAMPLE, INITIALIZATION SEGMENT
0004                       EXT [SYMT,[MAXS
0005                       EXT LU1
0006                       EXT EXEC,MPINT
0007                        EXT SEGLD,LIMEM
0008  00000 000001  D1     DEC 1
0009  00001 000002  D2     DEC 2
0010  00002          MPASI  EQU *
0011  00002 016007X        JSB LIMEM      CALCULATE AVAILABLE MEMORY
0012  00003 000007R        DEF *+4
0013  00004 000045R        DEF WHICH
0014  00005 000044R        DEF FWAM
0015  00006 000046R        DEF NWORD
0016*
0017*     CHANGE THE ALGOL ARRAY DESCRIPTOR FOR THE SYMBOL TABLE
0018*     ARRAY SO ITS LENGTH AND ADDRESS AND ``MAXSYMBOLS'' REFLECT
0019*     THE SIZE OF AVAILABLE MEMORY.
0020*
0021*
0022*
0023  00007 066033R        LDB SYMT       GET ADDRESS OF THE SYMBOL TABLE
```

```
0024  00010 005265          RBL,CLE,ERB   REMOVE THE INDIRECT
0025  00011 002041          SEZ,RSS       INDIRECT?
0026  00012 026015R         JMP *+3       NO, CONTINUE
0027  00013 164001          LDB B,I       YES, GET NEXT IN CHAIN
0028  00014 026010R         JMP *-4       AND KEEP LOOPING TILL INDIRECT IS GONE
0029*
0030*
0031  00015 076033R         STB SYMT      STORE DIRECT ADDRESS TO SYMBOL TABLE
0032  00016 046053R         ADB =D3       GET DESCRIPTOR WORD 4 ADDRESS
0033  00017 076047R         STB SIZ2      STORE ADDRESS OF 2ND DIMENSION SIZE
0034  00020 046054R         ADB =D2       ADVANCE TO ARRAY ADDRESS PNTR
0035  00021 062044R         LDA FWAM      GET FWA OF LARGEST SEGMENT
0036  00022 170001          STA B,I       ARRAY WILL GO IN HIGH MEMORY.
0037  00023 072043R         STA SSS         AND STORE ARRAY ADDRESS.
0038  00024 006400          CLB
0039  00025 062046R         LDA NWORD
0040  00026 100400          DIV SIZ2,I    CALCULATE # SYMBOLS
      00027 100047R
0041  00030 172035R         STA MAXS,I    STORE # SYMBOLS
0042  00031 016005X         JSB MPINT     CALL ALGOL PROCEDURE, FOR MAIN CODE EXEC
0043  00032 000037R         DEF RTRN
0044  00033 000001X SYMT    DEF [SYMT
0045  00034 000003X         DEF LU1
0046  00035 000002X         DEF [MAXS
0047  00036 000043R         DEF SSS
0048  00037          RTRN   EQU *
0049  00037 016006X         JSB SEGLD     FINISHED PASS 1, LOAD SEGMENT
0050  00040 000043R         DEF *+3         FOR PASS 1
0051  00041 000050R         DEF SEG1      FOR PASS 1
0052  00042 000044R         DEF FWAM
0053*
0054*     ******* NO RETURN FROM THIS ↑ ↑ ↑ CALL ↑ ↑ ↑ ↑
0055*
0057  00000          A      EQU 0
0058  00001          R      EQU 1
0059  00043 000000   SSS    NOP           STORAGE FOR ARRAY ADDRESS
0060  00044 000000   FWAM   NOP
0061  00045 000000   WHICH  NOP
0062  00046 000000   NWORD  NOP
0063  00047 000000   SIZ2   NOP
0064  00050 046520   SEG1   ASC 3,MPAS1
      00051 040523
      00052 030440
      00053 000003
      00054 000002
0065                        END MPASI
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
001 00000 HPAL,L,P,"MPTNT"
002 00000 &     SEGMENTATION EXAMPLE, INITIALIZATION
003 00000 PROCEDURE MPINT(SYMBOLTABLE,LU,MAXSYMBS,SYMBAD);
004 00001 INTEGER ARRAY SYMBOLTABLE;
005 00001 INTEGER LU,SYMBAD,MAXSYMBS;
006 00001 BEGIN
```

```
007 00007  WRITE(LU,#SYMBOLS ="I5", STARTING ADDRESS="K6),
008 00041              MAXSYMBS,SYMBAD);
009 00046 ;END;

PROGRAM= 000050  ERRORS=000


0001                  ASMB,R,L
0002  00000                 NAM MPAS1,5 SEGMENTATION EXAMPLE    ,PASS 1 CONTROL
0003*     EX7
0004*
0005                        EXT EXEC,MPPS1
0006                        EXT SEGLD
0007                        EXT MSG1,LU1
0008  00000          MPAS1  EQU *
0009  00000 016002X         JSB MPPS1      CALL ALGOL PROCEDURE, FOR MAIN CODE EXEC
0010  00001 000004R         DEF RTRN
0011  00002 000004X         DEF MSG1
0012  00003 000005X         DEF LU1
0013  00004          RTRN   EQU *
0014  00004 016003X         JSB SEGLD      FINISHED PASS 1, LOAD PASS 2
0015  00005 000007R         DEF *+2
0016  00006 000007R         DEF SEG2
0017  00007 046520   SEG2   ASC 3,MPAS2
      00010 040523
      00011 031040
0018                        END MPAS1
**  NO ERRORS *TOTAL **RTE ASMB 760924**


001 00000 HPAL,P,L,"MPPS1"
002 00000 &      EX2
003 00000 &
004 00000 &      SEGMENTATION EXAMPLE SEGMENT # 1
005 00000 &      ILLUSTRATING FIRST SEGMENT PROCESSING.
006 00000 &
007 00000 &
008 00000 &
009 00000 &
010 00000 PROCEDURE MPPS1(MSG1,LU1);
011 00001 INTEGER MSG1,LU1;
012 00001 &
013 00001 BEGIN
014 00005  PROCEDURE EXEC4(I1,I2,I3,I4);
015 00007  VALUE I1,I4;
016 00007  INTEGER I1,I2,I3,I4;
017 00007  CODE;
018 00005 &
019 00005  EXEC4(2,LU1,MSG1,7);
020 00040 &
021 00040  END;  & END OF PROCEDURE.

PROGRAM= 000042  ERRORS=000
```

```
0001                    ASMB,R,L
0002  00000                     NAM MPAS2,5 SEGMENTATION EXAMPLE    ,PASS 2 CONTROL
003*     EX6
0004*
0005                            EXT EXEC,MPPS2
0006                            EXT MSG2,LU1
0007  00000           MPAS2 EQU *
0008  00000 016002X           JSB MPPS2      CALL ALGOL PROCEDURE, FOR MAIN CODE EXEC
0009  00001 000004R           DEF RTRN
0010  00002 000003X           DEF MSG2
0011  00003 000004X           DEF LU1
0012  00004           RTRN  EQU *
0013  00004 016001X           JSB EXEC       FINISHED PASS 2, TERMINATE
0014  00005 000007R           DEF *+2
0015  00006 000007R           DEF D6
0016  00007 000006  D6        DEF 6
0017                            END MPAS2
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
001 00000 HPAL,P,L,"MPPS2"
002 00000 &      EX3
003 00000 &
004 00000 &      SEGMENTATION EXAMPLE SEGMENT # 2
005 00000 &      ILLUSTRATING SECOND SEGMENT PROCESSING.
006 00000 &
007 00000 &
008 00000 &
009 00000 &
010 00000 PROCEDURE MPPS2(MSG2,LU1);
011 00001 INTEGER MSG2,LU1;
012 00001 &
013 00001 BEGIN
014 00005  PROCEDURE EXEC4(I1,I2,I3,I4);
015 00007  VALUE I1,I4;
016 00007  INTEGER I1,I2,I3,I4;
017 00007  CODE;
018 00005 &
019 00005  EXEC4(2,LU1,MSG2,7);
020 00040 &
021 00040  END;  & END OF PROCEDURE.

PROGRAM= 000042  ERRORS=000
```

```
RMAIN   46002 46040  SEGMENTATION EXAMPLE,RESIDENT MAIN


MPASI   46041 46115  SEGMENTATION EXAMPLE, INITIALIZATION SEGMENT
SEGLD   46116 46307  RTE-II/III INTERFACE 4/12/76
EXEC4   46310 46326  3/16/76
MPINT   46327 46376
```

```
COR.A   46377 46412   92001-16005 741120
.PRAM   46413 46522   750701   24998-16001
FMTIO   46523 50132    24998-16002 REV.1610 760301
FRMTR   50133 52734    24998-16002 REV.1610 760301
FMT.E   52735 52735    24998-16002 REV.1610   760301
RETO    52736 53040   92001-16005 741120
.OPSY   53041 53100   750701   24998-16001

MPAS1   46041 46052   SEGMENTATION EXAMPLE    ,PASS 1 CONTROL
MPPS1   46053 46114

.PRAM   46115 46224   750701   24998-16001
SEGLD   46225 46416   RTE-II/III INTERFACE 4/12/76
EXEC4   46417 46435   3/16/76

COR.A   46436 46451   92001-16005 741120



MPAS2   46041 46050   SEGMENTATION EXAMPLE    ,PASS 2 CONTROL
MPPS2   46051 46112

.PRAM   46113 46222   750701   24998-16001
EXEC4   46223 46241   3/16/76


 4 PAGES REQUIRED


SYMBOLS = 446, STARTING ADDRESS=053101
MESSAGE FROM ♦
TWO SEGMENTS.
```

# LISTING DMS MAP REGISTERS ON-LINE IN RTE

*Larry W. Smith/DSD*

In order to better understand how the DMS system relates to RTE-III, a picture of the actual contents of all DMS registers 0-377 would be helpful to visualize memory layout. The following assembly subroutine named 'MAPMX' illustrates how to retrieve a set of 32 DMS map registers (System, Port A, Port B, or User) and print them with a second test routine written in FORTRAN IV. Unfortunately, the instructions which store the appropriate maps into memory generate a DMS interrupt causing RTE to abort the calling program. Thus, the routine 'MAPMX' must be coded as privileged. Compare the output of WHZAT's partition breakdown to the output of the test routine 'MAP'.

This subroutine will retrieve and return the appropriate DMS registers:

```
            ASMB,R,L *** LIST DMS MAPS ***

00000               NAM MAPMX,7

00000                    ENT MAPMX

                    EXT ENTR,$LIBR,$LIBX

 DESCRIPTION
 -----------

  THIS SUBROUTINE RETURNS THE 32 MAP REGISTERS OF THE SYSTEM,
  PORT A, PORT B, OR USER MAP.

 CALLING SEQUENCE
 ---------------

   CALL MAPMX(ITYPE,IARRY)

     ITYPE ---> TYPE OF MAP:  ITYPE=2HA - PORT A
                              ITYPE=2HB - PORT B
                              ITYPE=2HU - USER
                              ITYPE=2HS - SYSTEM
     IARRY ---> ARRAY ADDRESS OF AT LEAST 32 WORDS

00000 000000 ITYPE NOP            TYPE OF MAP TO RETURN.
00001 000000 IARRY NOP            ADDRESS FOR 32 MAP REGISTERS.
00002 000000 MAPMX NOP            < ENTRY/EXIT >

00003 016001X      JSB .ENTR
00004 000000R      DEF ITYPE

00005 062033R      LDA 3B100000   SET FOR MAP LOAD INTO MEMORY.
00006 032001R      IOR IARRY      MERGE IN ADDRESS IN BITS 0-14.
00007 166000R      LDB ITYPE,I    GET MAP TO LIST.

00010 016002X      JSB $LIBR      MUST GO PRIVELEGED TO AVOID DM ERROR.
00011 000000       NOP
00012 056034R      CPB =AS        SYSTEM MAP?
00013 026026R      JMP SYSTM
00014 056035R      CPB =AA        PORT A MAP?
00015 026022R      JMP PORTA
00016 056036R      CPB =AB        PORT B MAP?
00017 026024R      JMP PORTB

00020 101711       USA            ASSUME USER 32 MAPS.
00021 026027R      JMP DONE

00022 101712 PORTA PAA            TRANSFER PORT A 32 MAPS.
00023 026027R      JMP DONE

00024 101713 PORTB PBA            TRANSFER PORT B 32 MAPS.
00025 026027R      JMP DONE

00026 101710 SYSTM SYA            TRANSFER SYSTEM 32 MAPS.
00027 016003X DONE JSB $LIBX
00030 000031R      DEF *+1
00031 000032R      DEF *+1
00032 126002R      JMP MAPMX,I

...LITERALS...
 00033 100000
 00034 051440
 00035 040440
 00036 041040

                    END
NO ERRORS *TOTAL **RTE ASMB 760924**
```

To test this routine, the following FORTRAN program could be used:

```
      FTN4,L
            PROGRAM MAP
            DIMENSION IARRY(32)
            CALL MAPMX(2HS ,IARRY)
            WRITE(6,100)IARRY
        100 FORMAT(///" SYSTEM MAP REGISTERS 0-37 (1-32)"//8(X,K6))
            CALL MAPMX(2HA ,IARRY)
            WRITE(6,101)IARRY
        101 FORMAT(///" PORT A MAP REGISTERS 40-77 (33-64)"//8(X,K6))
            CALL MAPMX(2HB ,IARRY)
            WRITE(6,102)IARRY
        102 FORMAT(///" PORT B MAP REGISTERS 100-137 (65-96)"//8(X,K6))
            CALL MAPMX(2HU ,IARRY)
            WRITE(6,103)IARRY
        103 FORMAT(///" USER   MAP RGISTERS 140-177 (97-128)"//8(X,K6))
            END

**  NO ERRORS**    PROGRAM = 00208      COMMON = 00000
```

The output of this routine which ran in partition #10 as in the partition of WHZAT below, is as follows:

System Map Registers 0-37 (1-32)

```
000000 000001 000002 000003 000004 000005 000006 000007
000010 000011 000012 000013 000014 000015 000016 000017
000020 000021 000024 000025 000026 000027 000030 000031
000032 000033 000034 000035 000036 100035 100036 100037
```

Port A Map Registers 40-77 (33-64)

```
000000 000001 000002 000003 000004 000005 000006 000007
000010 000011 000012 000013 000014 000015 000016 000017
000020 000021 000024 000025 000026 000027 000030 000031
000032 000033 000034 000035 000036 100035 100036 100037
```

Port B Map Registers 100-137 (65-96)

```
000067 000001 000002 000003 000004 000005 000006 000007
000010 000011 000012 000013 000014 000015 000016 000017
000020 000021 000070 000071 000072 000073 000074 140000
140001 140002 140003 140004 140005 140006 140007 140010
```

User Map Registers 140-177 (97-128)

```
000221 000001 000002 000003 000004 000005 000006 000007
000010 000011 000012 000013 000014 000015 000016 000017
000020 000021 000222 000223 000224 140000 140001 140002
140003 140004 140005 140006 140007 140010 140011 140012
```

The program was run under the following system (RU,WHZAT,,1):

```
 9:18:58:420
*********************************************************
PTN# SIZE   PAGES   GB/RT PRGRM
*********************************************************
 1     4    31-  34    RT (NONE)
 2     5    35-  39    RT (NONE)
 3    15    40-  54 BG RT (NONE)
 4    15    55-  69 BG    (NONE)
 5    15    70-  84 BG    (NONE)
 6    15    85-  99 BG    (NONE)
 7    15   100- 114 BG    (NONE)
 8    15   115- 129 BG    (NONE)
 9    15   130- 144 BG    (NONE)
10    15   145- 159 BG    MAP
*********************************************************
 9:18:58:820
```

If you are implementing your own memory management scheme while running RTE, this routine together with the information in the article "Returning RTE-III Memory Size in Number of Pages", might help you.


In this issue I would like to share with you programming tips I have collected on the IMAGE/1000 data base management software. Also I will correct the usage of IGET as shown in this column in the previous issue. I would like to thank ***Bob Funk***, HP Systems Engineer, Richardson, Texas, for his inputs on IMAGE.

What are the considerations in determining the size and structure of a data base application program? How can one optimize the file manager overhead spent on locating the files? What is required by DBINT? When is an add or delete posted to the disc? How can a user determine the percentage completed at any point in the execution of a long program?

## SIZE AND STRUCTURE OF AN APPLICATION PROGRAM

The data base management system (DBMS) consists of eleven subroutines which provide the user with the capability of opening a data base; reading, writing, and updating data items; retrieving information about the data base structure; locking and unlocking a data base; and closing a data base. DBMS requires a work area in memory for the run table and buffer space for transfer of data items to and from the disc. For a large application then, segmentation becomes a necessity. As a warning remember that it is easier to initially structure a program into segments than to rewrite a completed unsegmented program which is too large.

In planning the structure of the program and function of each of the modules it is worth the time spent in first mapping out the partition space to determine the amount of available programming area for the main and segments. From the total area subtract the size of the run table, the maximum desired data control block size (DCB) for the data set files, and, if possible, the size of the main. The remaining value then, is the length which each segment should not exceed at load time. The file DCB's are allocated in one of four possible combinations of sizes:

1. six DCB's of 272 words each.
2. six DCB's of 144 words each.
3. one DCB of 272 words.
4. one DCB of 144 words.

As a result, a small increase in program size could have a dramatic effect on program performance. The user should initially try to limit the program to 80% of the available space, thus allowing for future modifications.

## OPTIMIZE FMGR OVERHEAD ON DBOPN

DBOPN opens the data base by transferring a copy of the root file into the run table. The cartridge reference number on which the root file is located is not given in the call and a complete search of the file system will be performed until the file is found. Placing the root file on LU2 will save some overhead here. The file opens for the data sets are handled differently. In the root file the sixth character of the file name is replaced with the cartridge reference number of the file. User needs only to place the data sets at the top of the directory list to save time spent by the system searching for the files on that cartridge. Remember that the file manager STORE, COPY, and DUMP commands will truncate records longer than 128 words so root and data set files should not be moved with these commands or information may be lost.

## DBINT REQUIREMENTS

The calling sequence for DBINT requests the number of segments accessing the data base followed by a list of the segment names. DBINT uses this information to allocate the space for the run table and file DCB's. DBINT requires only the name of the main and the largest segment. It is cleaner programming to pass DBINT the complete list of segment names and then use an index into the buffer to retrieve the names when needed, for example, on the EXEC request to load a segment into memory.

## POST OF DCB'S AND RUN TABLE

To avoid the loss of valuable information in memory buffers after an update, put, or delete operation, a programmer should be aware of when data is actually written to the disc from the run table and the DCB's. Let's consider first the case of a data base open in mode three.

When the put, delete, or update is executed the file manager uses the DCB to buffer the write or read to the file. The data is not written to the file until the buffer is filled or needed for a


read of data not currently in the buffer. DBCLS posts the run table to the root file and posts any data remaining in the DCB which needs to be written. As a result, if a program which has performed puts, updates or deletes terminates before the execution of the DBCLS the remaining data in the DCB's and the run table will be lost.

In mode two the update operation is the same as above, but the put and delete are handled differently. A call to DBLCK is required to set up the resource number to lock the data base. The put and delete retrieve the volatile data, post the DCB's, perform the desired put or delete, write the volatile data back to memory, and again post the DCB's. When the user relinquishes exclusive access to the data base with the call to DBULK the resource number is returned to the system. The run table data is posted to the root file when the last user open in mode two executes the call to DBCLS If a program terminates without the call to DBCLS the utility RECOV may be used to simulate closure of the data base, that is to decrement by one the count of users open in mode two and post the data to the root file if the count of users is now zero.

## PERCENTAGE COMPLETED IN EXECUTION OF LONG PROGRAM

For a long program, for example, one which dumps the entire contents of the data base to mag tape, it is convenient to use the DOS/RTE relocatable library function IFBRK. The function allows the user to break into execution of the program by checking for the value of the break flag. In this manner the programmer may, upon a break, return the percentage of the task completed at the current time. The FORTRAN calling sequence for IFBRK is:

```
IF (IFBRK(IDUMY)) 10,20
```

Where:

10 branch will be taken if the break flag is set. The break flag will be cleared.

20 branch will be taken if the break flag is not set.

## CORRECTION ON USAGE OF FUNCTION IGET

The program LIBLS printed in issue 12 uses the technique of retrieving base page values by dimensioning an array IGET and setting IGET(0) to one. Please note that this will not work with the new FORTRAN revision 1726. The relocatable library routine IGET may be used for the same purpose.

IF you have any questions, suggestions, or comments about your HP 1000 (9600) system, please address them to:

SOFTWARE SAMANTHA
c/o Communicator 1000(9600) Group
HP Data Systems Division
11000 Wolfe Road
Cupertino, CA. 95014

# WRITING PROGRAMS TO USE EITHER FILES OR DEVICE LU's

*Jim Bridges/DSD*

It is frequently desirable to specify either a physical device (e.g., mag tape) or a file name for I/O. Perhaps the program converts lower case ASCII characters to upper case characters. In one part of the program, the user might be prompted with:

## OUTPUT (FILE NAME OR DEVICE LU)?:

The upper case text will go to a line printer (for example) or a file. The main body of the program can be coded such that it does not matter whether the output is to an LU or to a file if type 0 files are used. This presents some additional problems, however. If an existing type 0 is to be used, then the person at the terminal must know the names (and associated devices) of those type 0 files. If the type 0 file does not exist, the program can not create one: this must be done through the program FMGR. The creation of a type 0 files causes the file directory to be re-organized such that the type 0 file appears at the top. This is an operation which sometimes consumes a great deal of time.

There is an alternative which uses the FMP (file management package) subroutines by building a proper DCB (data control block) for use by the program only. That is, the program uses the features of a type 0 file but no entry for the type 0 file is made in the directory. This is accomplished by initializing a DCB according to the information on page C-2 of the appendix of the BSM (Batch Spool Monitor) manual (92060-90013). Not all elements of the DCB need be filled in. An example appears below:

```
FTN4,L
      PROGRAM SAMPL
      INTEGER DCB (16)
C NOTE THAT THE BUFFER AFTER THE 16 WORDS WILL
       NOT BE USED
      ......
      ......
      DCB (1) = 177400B
      DCB (2) = 0
      DCB (3) =0
      DCB (4) = LU
      DCB (5) = IFUNC+LU
      DCB (6) = 100001B
      DCB (7) = 100001B
      DCB (8) = 100000B
      DCB (9) = 0

C  NOTE: DCB (10) SET TO ID SEGMENT ADDRESS OF
         THIS PROGRAM

      DCB (10) = IGET (1717B)
      DCB (11) = 0
      DCB (12) = 0
      DCB (13) = 0
      DCB (14) = 0
      DCB (15) = 1
      DCB (16) = 0
      ......
```

LU is the device logical unit. IFUNC is the end-of-file code described on page C-3: the EOF code is chosen according to the actual device. Three codes are given as follows:

```
0100B (+ LU)    EOF on mag tape
1000B (+ LU)    EOF on paper tape
1100B (+ LU)    EOF on line printer
```

In order to select the proper function, the program needs to read the octal code for the driver type from the EQT table associated with that LU. A method of doing this is:

```
IDRT = IGET (1652B)
IEQT = IGET (IDRT + LU - 1)
IEQT1 = (IEQT-1)*15 + IGET (1650B)
IVALS = IGET (IEQT1 + 4)
IDVR = (IAND (IEQT5,77400B))/256
```

The value in IDVR is an octal number corresponding to the driver type, e.g., 23B for DVR23 (mag tape driver). If the driver type is not 23B, 02B or 12B (corresponding to the EOF codes above) then the program might select the line printer function for a terminal (type 00B or 05B): the driver may or may not process the code if it is not any of the above types.

Once the DCB has been set up, the program can call all of the standard FMP subroutines, such as OPEN, CLOSE, READF and WRITF.

The reader should study the information on pages C-2 and C-3 in conjunction with this article.

## HP 7905 DISC MAPPING AID FOR RTGEN

*Joe Bailey/NSC*

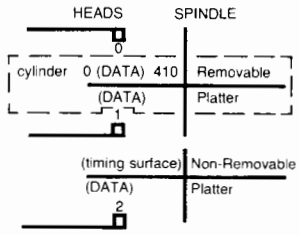
Each time you build an RTE system on the HP 7905, you are requested to answer a series of disc configuration (or *logical to physical* mapping) questions, which can almost always be perplexing unless you happen to have a 3 by 411 matrix in your head. This note, then, will help the rest of us unfortunates without that spare matrices capability.

Please refer to the RTE-II Prog. & Oper. Manual 92001-93001 pp 6.9 HP 7905 Disc Configuration, or the RTE-III Prog. & Oper. Manual 92060-90004 pp 6.6 HP 7905 Disc Configuration for the basic approach to this problem.

First some definitions:

- An HP 7905 Disc Drive has 3 data heads and 3 data surfaces.
- A surface has 411 cylinders.
- A track is defined as one head on one cylinder for one revolution. A track contains 48 each 128 word addressable sectors. RTE thinks that all sectors have 64 words each, or 1/2 of one HP 7905 physical sector.
- A disc channel is the interface card which may have up to eight HP 7905 drives (16 platters or 12 surfaces of 411 cylinders, each). Each drive consists of 1233 tracks to be allocated contiguously to (logical) subchannels, each assigned a unique DRT # (LU#).
- A (logical) subchannel is a collection of contiguous tracks.
- Spare tracks (a hardware feature, software implemented) are allocated at the end of a logical subchannel by the user at RTGEN time. They are implemented by the RTGEN or Disc Backup Facility when a 'bad track' is encountered. A 'bad' track is one that cannot be read as it was written.

So, we have the following structure:

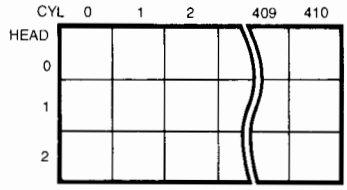



which translates into the matrix:

VOLUME(HEAD,CYLINDER)

where: HEAD 0,1,2

CYLINDER 0, thru 410




or an integer array:

VOLUME (0:2, 0:410)

or an integer array:

VOLUME (0:2, 0:410)

In mapping this creature, the following considerations must be taken into account:

1. A system disc must have ≤256 tracks, an RTE system disc ID segment limitation.

2. A system subchannel has S tracks for the absolute system code, F tracks for file management package (FMP) N tracks for general usage, for a total of T tracks. Spare tracks are not addressable directly, and are external to the logical subchannel.

   S is generally 20-40.

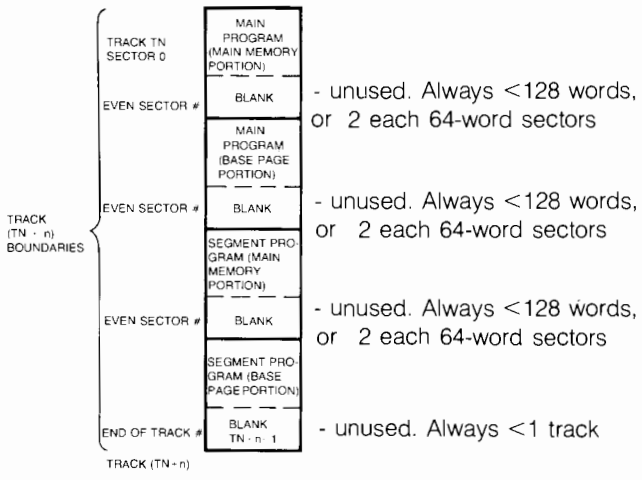
   N is what is left over for on-line loads, edits, FMP PACK, RTGEN.

   F is generally 100.

   T must be ≤256 = S + F + N.

   It is recommended that:

   - N be large enough to permit two complete copies of the largest expected file to be edited (an experience factor)
   - On-line loads, not replacing an RTGEN'ed program, should be temporarily loaded, then save program'd thru FMGR and then deleted from the system by 'OF, <Program name> ,8
   - An FMGR pack command requires that N be ≥8 tracks on the system logical subchannel (LU 2).
   - RTGEN (on-line) requires 6 tracks be allocated from N for its virtual memory scheme.
   - ON-line loaded and swapped programs (requiring n tracks) will be stored into contiguous tracks, as follows:




   Read or writes to these areas require:

   1) Time to access starting track for Main program, plus read/write time, plus
   2) Time to reaccess spared track, if applicable, plus
   3) Time to access second track of Main program, if main crosses track boundary, plus read/write time, plus
   4) Time to reaccess spared track, if applicable, plus
   5) Time to access Base Page portion, which can be a repeat of 1 thru 4 above, plus read/write time.
   6) Time to access Segment program when scheduled, which can be a repeat of 1-4 above, plus
   7) Repeat of step 5

3. An auxiliary logical subchannel (LU3) must have ≤ 256 tracks.

4. An auxiliary disc is not required, but is sometimes necessary:
   - for large file edits
   - for more type 6 (SAVE PROGRAM) files
   - to decrease swapping time, since system tracks are allocated from the top of the available track list, from last available track towards first available track in contiguous chunks. This feature permits the auxiliary disc to be used as a "swapping disc", since LU3 can be another controller and disc, this is the ultimate step in optimizing a system for speed.

5. A peripheral logical subchannel for use as an FMP cartridge must have ≤ 1024 tracks, since this is the limit of addressability of the FMP subsystem.

6. Logical subchannels may be mapped in cylinder or surface mode. If cylinder mode is selected, either 2 or 3 heads may be used since 1 head implies surface mode. Heads are allocated (in cylinder mode) from lowest to highest (e.g.: if the starting head is 0, and 3 surfaces are specified, then heads 0, 1, and 2 will be used, in that order).

   Maximum head travel per logical subchannel is equal to: ENTIER [(number of tracks per subchannel) divided by (the number of heads per subchannel)].

*Example #1*

\# heads = 2
& # tracks = 203

then ENTIER (203/2) = 102

- 102 cylinders must be transversed to move from track 0 to track 202

  and

- 2 tracks may be accessed without head movement

*Example #2*

*If* # heads = 3
& # tracks = 203

then ENTIER (203/3) = 68

- 68 cylinders must be transversed to move from track 0 th track 202

  and

- 3 tracks may be accessed without head movement. But, the rotational alignment between the two platters (removable and non-removable) in one drive, depends on drive orientation when the platter is inserted. This single factor makes the track-to-track access time across platters unpredictable.

7. Where a user has a mag tape, it is recommended that the 7905 be mapped in cylinder mode, with 2 or 3 heads per subchannel, since it is quicker to overlay an FMP subchannel from mag tape than to replace the removable pack. Also, changing the packs will introduce dust into an otherwise clean environment. By using the mag tape, you also achieve an off-line backup.

8. Where 7905 and 7900 are in use at the same locations, it is recommended that the 7905 be mapped as 203 tracks per subchannel, in either cylinder or surface mode. This permits interchangeability of FMP Logical Cartridges via mag tape.

9. RTE and the 7905 have an automatic track sparing feature in case 'bad track' is encountered during RTGEN. Track sparing will be set up by RTGEN, or the disc back up facility (in '*unit mode*' only). To use this feature, some tracks must be reserved (not assigned to any logical subchannel), and designated as 'spare' tracks for a particular logical subchannel. Spare tracks are allocated at the end of the logical subchannel for *that* subchannel! The operator is notified of any sparing during RTGEN or Disc Backup facility restore process. When a 'BAD' track is addressed that was 'spared', information on that 'BAD' track causes the hardware to re-address to the sparing track which is to be found at the end of that logical subchannel. This, of course, will create an extra seek, and the time added to the original seek time for that track. It may be desirable to replace that platter or, attempt a bulk erase and reformat to regain the 'BAD' track. (IOSM 7905A-0576-02).

10. There will be only 400 guaranteed tracks/surface, with 6 of these reserved for spares. (IOSM 7905-0576-02).

11. A maximum of 5 defective tracks per surface is allowable. (IOSM 7905A-0576-02).

12. HP 7905 cartridges shipped from the factory will have no more than 3 defective tracks out of 411 on each surface of the cartridge. These will be listed on the bottom of the cartridge. HP 7905 cartridges shipped are already formatted. If used in several drives, they should be bulk erased and reformatted from time to time. (IOSM 7905A-0576-02).

Before attempting to arrive at an approach, consider how the system is to be used by its users.

If the users are new to this type of system, they will generally be convinced that all file data they will ever create will fit within the on-line accessible disc space, and no off line data storage will ever be necessary. After some experience, an off line storage technique becomes imperative.

The question becomes — "mag tape or disc platter?" Using the Disc Backup Utilities provided, of course. Well, for most users, both methods usually requires a high entry fee since he must have two disc drives or one disc drive and a mag tape. One 7905 drive can be stored on one 1200 foot reel of tape at 800 BPI in 10 to 15 minutes at 40 IPS, and at a cost of about $20.00 U.S.

One 7905 drive can be stored on two 7905 platters in about 4 minutes at a cost of about $300.00 U.S.

Restoring, in either case, permits reformatting and track sparing.

What does all this have to do with disc mapping, you ask. Well, if the user selects the disc backup, he should have his file space on the removable disc so that he can interchange platters, take the risk of introducing dust into an otherwise clean environment, and take the time to stop and restart the disc while disabling access to the non-removable platter during that time. This mapping forces the system to the non-removable platter, mapped in surface mode (the heads must move on every track access) and will slow the system down when loading or swapping a program. On the other side, if the user selects mag tape backup, he can map his disc as he

pleases, optimizing system speed by employing cylinder mode (if two or three cylinder mode, two or three consecutive tracks can be accessed without moving the heads). In this case, he does not intend to *ever* remove the platter, the disc stays clean, but he must be aware of mag tape break-in and deterioration problems.

During RTGEN in the initialization phase, you are asked:

> "# TRKS, FRST CYL #, HEAD #, # SURFACES, UNIT, # OF SPARES FOR SUBCHANNEL:"

**# OF TRACKS::** = how many tracks on that subchannel (see considerations 1,2,5,8)

**FIRST CYLINDER #::** = the starting cylinder for this subchannel. The first cylinder # for the first subchannel can be determined by the operators choice. The next cylinder # becomes a problem solvable by the following equation:

> NEXT CYL # = previous cyl # + ENTIER {(# of tracks/subchannel)/)# of surfaces/subchannel)} (Just be sure that you don't exceed cyl #410.)

**HEAD #::** = the starting head # for this subchannel. The Head # can be difficult to determine. If it is the first subchannel, it is at your discretion. Otherwise, the next head available is:

> HEAD # = { [previous HEAD # + (# of tracks/subchannel) MODULO (# of surfaces/subchannel) −1] MODULO (# of surfaces/subchannel)} +1

**# OF SURFACES::** = if equal to 1 then surface mode, else cylinder mode. (see considerations)

**UNIT::** = drive number. Determined by hardware cabling.

**# OF SPARES FOR SUBCHANNEL::** = (See considerations 9, 10, 11, 12).

NOTE

> Off-line RTGEN will reinitialize (and wipe out) the system and auxiliary subchannels, totally. On-line RTGEN does not reinitialize any disc tracks, but SWTCH does modify either the system portion or all of the system and auxiliary subchannels, as directed by the operator.

*EXAMPLE #1*

Map a 7905 disc into 6 logical subchannels of 203 tracks each in cylinder mode; 3 heads per cylinder. Each logical subchannel is to begin at head 0.

*1st Response* (is at the users discretion.) (System Subchannel)

203,0,0,3,0,??
??= # of spares, or # of heads to complete cylinder

NEXT HEAD # = ((0+((203)MOD( 3))−1)MOD 3)+1
= ((0+2−1)MOD 3)+1
= 2

?? = 1,@ 3/cylinder (to use complete cylinders)
?? = 1, 4, 7, 10, etc.

and the 1st response is:

203,0,0,3,0,1

| HEAD | 0 | 1 | 2 | 3 | 66 | 67 | 68 | 69 | 410 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 6 | 9 | 199 | 201 | — | — | — |
| 1 | 1 | 4 | 7 | 10 | 199 | 202 | — | — | — |
| 2 | 2 | 5 | 8 | 11 | 200 | S | — | — | — |

*2nd Response* (must be calculated) (Auxiliary subchannel)

203,?,0,3,0,??
? = 0 + ENTIER ((203 + 1)/ 3)
= 0 + ENTIER (68)
= 68
?? = the same as in response #1 and the second response is:
203,68,0,3,0,1

| HEAD | 0 | 1 | 67 | 68 | 69 | 134 | 135 | 136 | 137 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 201 | 0 | 3 | 198 | 201 | — | — | — |
| 1 | 1 | 4 | 202 | 1 | 4 | 199 | 202 | — | — | — |
| 2 | 2 | 5 | S | 2 | 5 | 200 | S | — | — | — |

*3rd Response* (must be calculated) (Spool subchannel)

203,?,0,3,0,??
? = 68 + ENTIER ((203 + 1)/(3))
= 136
?? = same as before and the third response is:

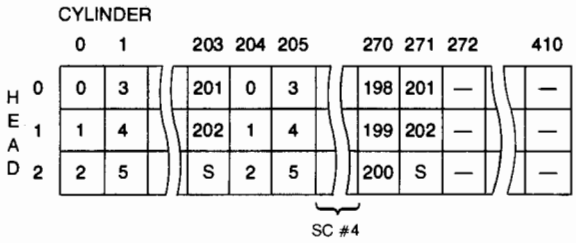
203,136,0,3,0,1

| | 0 | 1 | 134 | 135 | 136 | 137 | 202 | 203 | 204 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 198 | 201 | 0 | 3 | 198 | 201 | — | — |
| 1 | 1 | 4 | 199 | 202 | 1 | 4 | 199 | 202 | — | — |
| 2 | 2 | 5 | 200 | S | 2 | 5 | 200 | S | — | — |

*4th Response* (must be calculated)

203,?,0,3,0,??
? = 136 + ENTIER ((203 + 1)/(3))
= 204
?? = same as before and fourth response is:

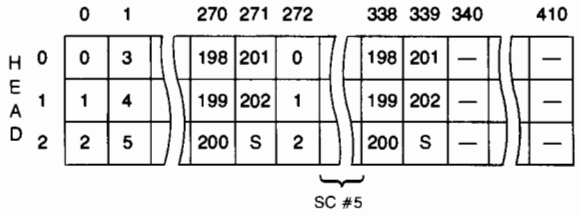
203,204,0,3,0,1




*5th Response* (must be calculated)

203,?,0,3,0,??
? = 204 + 1)/(3))
= 272
?? = same as above and the fifth response becomes:

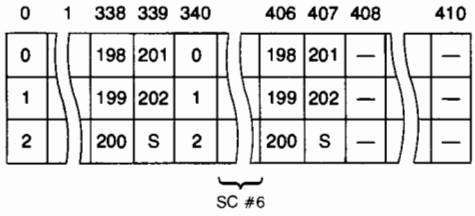
203,272,0,3,0,1




*6th Response* (must be calculated)

203,?,0,3,0,??
? = 272 + ENTIER ((203 + 1)/(3))
= 340
?? = same as above and the sixth response becomes:

203,240,0,3,0,1




*CHECK*

Have we exceeded cylinder 410?

? = 340 + ENTIER ((203 + 1)/(3))
= 408

Answer is NO and we find three cylinders (408, 409, 410) and 9 tracks unused.

According to the DMD Lab, we shouldn't use the last cylinder; so that allows us to assign 6 of the 9 unused tracks. To increase the longevity of the system and auxiliary subchannels, let us add 3 spares each to those subchannels, which will bump the starting cylinder # for the following subchannels by one each. Our responses become:

1st Test
Subchannel #0 (SYSTEM)
203,0,0,3,0,(1 + 3)

2nd Test
Subchannel +1 (AUXILIARY)
203, (68 + 1), 0,3,0, (1 + 3)

3rd Test
Subchannel #2 (SPOOL)
203, (136 + 2), 0,3,0,1

4th Test
Subchannel #3
203, (204 + 2), 0,3,0,1

5th Test
Subchannel #4
203, (272 + 2), 0,3,0,1

6th Test
Subchannel #5
203, (304 + 2), 0,3,0,1

Now for the final check:

? = 342 + ENTIER ((203+1)/3) = 342 + 68
= 410

The next available cylinder is #410 and we are not using it! As a further check, there are 411 cylinders and 3 heads, which gives 3×411 = 1233 Tracks. We have 6 logical subchannels of 203 tracks each, and 12 spares, which gives 6×203+12 = 1230 Tracks (leaving the three of cylinder 410 unused, as required).

And the final matrix looks like:

| | 0 | 67 | 68 | 69 | 136 | 137 | 138 | 204 | 206 | 273 | 274 | 275 | 340 | 341 | 342 | 408 | 409 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 201 | S | 0 | 201 | S | 0 | 201 | 0 | 201 | 0 | 3 | 198 | 201 | 0 | 198 | 201 | — |
| 1 | 1 | 202 | S | 1 | 202 | S | 1 | 202 | 1 | 202 | 1 | 4 | 199 | 202 | 1 | 199 | 202 | — |
| 2 | 2 | S | S | 2 | S | S | 2 | S | 2 | S | 2 | 5 | 200 | S | 2 | 200 | S | — |
| | | SC #0 | | | SC #1 | | | SC #2 | | SC #3 | | | SC #4 | | | SC #5 | | |

*EXAMPLE #2*

Map the removable platter in 203 tracks per logical subchannel in 2 head/cylinder mode. Map the non-removable platter in 203 tracks per logical subchannel in 1 head per cylinder (surface) mode. Each subchannel on the removable platter is to begin at head 0.

*Non Removable Platter Mapping*

*Response #1* (arbitrary) System

203,0,2,1,0,2 (System Disc)

| | 0 | 1 | 201 | 202 | 203 | 204 | 205 | 206 | 410 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | — | — | — | — | — | — | — | — | — |
| 1 | — | — | — | — | — | — | — | — | — |
| 2 | 0 | 1 | 201 | 202 | S | S | — | — | — |

*Response #2* (calculated from above) Auxiliary

203,?,2,1,0,2
? = 0 + ENTIER ((203 + 2)/(1))
= 0 + ENTIER (205)
= 205

and the response +2 becomes:

203,205,2,1,0,2

CYLINDER

| | 0 | 202 | 203 | 204 | 205 | 206 | 407 | 408 | 409 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | — | — | — | — | — | — | — | — | — | — |
| 1 | — | — | — | — | — | — | — | — | — | — |
| 2 | 0 | 202 | S | S | 0 | 1 | 2 | 202 | S | S | — |

*Check for non-removable platters*

It has 411 cylinders and 1 head = 411 tracks. We used 2 logical subchannels of 203 tracks each, and 4 spares, 203×2+4=410 tracks. Also, have we exceeded cylinder #409 in response #2?

? = 205 + ENTIER ((203 + 2)/(1))
= 205 + ENTIER (205)
= 410

No, we will not be using next cylinder #410!

*Response #3* (arbitrary) FMP Cartridge

203,0,0,2,0,??
?? = # of heads to complete cylinder
next head # = ((0 + ((203) MOD (2))−1)MOD 3)+1
= (0 + 1 − 1) MOD 3) −1
= 1

since we are only using heads 0 and 1,
?? = 1
and our response becomes:
203,0,0,2,0,1

CYLINDER

| | 0 | 1 | 100 | 101 | 102 | 409 | 410 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 200 | 202 | — | — | — |
| 1 | 1 | 3 | 201 | S | — | — | — |
| 2 | 0 | 1 | 100 | 101 | 102 | S | — |

*Response #4* (must be calculated)

203,?,0,2,0,??
? = 0 + ENTIER ((203 + 1)/(2))
= 102

?? same as before and our response becomes:203,102,0,2,0,1

| | 0 | 100 | 101 | 102 | 103 | 202 | 203 | 204 | 205 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 200 | 202 | 0 | 2 | 200 | 202 | — | — | — |
| 1 | 1 | 201 | S | 1 | 3 | 201 | S | — | — | — |
| 2 | 0 | 100 | 101 | 102 | 103 | 202 | S | S | 0 | — |

*Response #5* (must be calculated)

203,?,0,2,0,??
? = 102 + ENTIER ((203 + 1)/(2))
= 204

?? same as before and our response becomes:

203,204,0,2,0,1

| | 0 | 202 | 203 | 204 | 205 | 304 | 305 | 306 | 307 | 308 | 409 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 200 | 202 | 0 | 2 | 200 | 202 | — | — | — | — | — |
| 1 | 1 | 201 | S | 1 | 3 | 201 | S | — | — | — | — | — |
| 2 | 0 | 202 | S | S | 0 | 98 | 99 | 100 | 101 | 102 | S | — |

*Response #6* (must be calculated)

203,?,0,2,0,??
? = 204 + ENTIER ((203 + 1)/(2))
= 306
?? same as before and our response becomes:

203,306,0,2,0,1

| | CYLINDER 0 | 304 | 305 | 306 | 307 | | 407 | 408 | 409 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 200 | 202 | 0 | 2 | 4 | 202 | — | — | — |
| 1 | 1 | 201 | S | 1 | 3 | 5 | | — | — | — |
| 2 | 0 | 98 | 99 | 100 | 101 | 102 | 201 202 | S | S | — |

*Check*

Have we exceeded cylinder 410?

? = 306 + ENTIER ((203 + 1)/(2))
= 408

No, and we have two cylinders of two heads remaining (408, 409), or a total of four tracks. (We shouldn't use cylinder 410, remember.) Because we have specified our subchannels to begin with head 0, we can't add one track to each subchannel, so let us add two tracks (one cylinder) each to subchannels #2 and #3 (the first two FMP subchannels). And we have 411 cylinders × 2 heads + 822 tracks available on the removable pack. We have declared 2 each 203 tracks with 3 spare logical subchannels (= 412 tracks) and 2 each 203 tracks with 1 spare logical subchannels (= 408 tracks) = 820 tracks total (remember cylinder 410 is not used).

The final response is,

S.C. #0
203,0,2,1,0,2

S.C. #1
203,205,2,1,0,2

S.C. #2
203,0,0,2,0,3

S.C. #3
203,103,0,2,0,3

S.C. #4
203,206,0,2,0,1

S.C. #5
203,308,0,2,0,1

| | 0 | 1 | 100 | 101 | 102 | 103 | 104 | 201 | 202 | 203 | 204 | 205 | 206 | 306 | 307 | 308 | 309 | 310 | 407 | 408 | 409 | 410 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 200 | 202 | S | 0 | 2 | 196 | 198 | 200 | 202 | S | 0 | 2 | 200 | 202 | 0 | 2 | 4 | 198 | 200 | 202 | — |
| 1 | 1 | 3 | 201 | S | S | 1 | 3 | 197 | 199 | 201 | S | S | 1 | 3 | 201 | S | 1 | 3 | 5 | 199 | 201 | S | S |
| 2 | 0 | 1 | 100 | 101 | 102 | 103 | 104 | 201 | 202 | S | S | 0 | 1 | 2 | 100 | 101 | 102 | 103 | 104 | 202 | S | S | S |

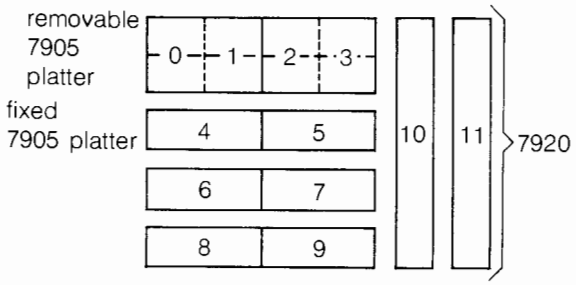
## MAPPING 7920 AND 7905 FOR SUB-CHANNEL COMPATIBILITY

*Gary Gubitz/DSD*

This is a technique for defining disc channels such that the system will be compatible with either a 7905 or 7920 disc drive. Therefore, if you are generating a 7905 system but are anticipating the acquisition of a 7920 you could configure your subchannels to avoid having to regenerate your system. When your 7920 arrived, you might use the off-line disc backup utility (!DSKUP) to save the system from the 7905 on magnetic tape and then when you replace the 7905 with the 7920, restore the back-up from the tape to the 7920 and your system, with all your programs and files, will be ready to go (please refer to the RTE Utility Programs Reference Manual, #92060-90017, for more information on using this utility).

| sub channel | #tracks | 1st cylinder # | head # | #surfaces | unit | #spares |
|---|---|---|---|---|---|---|
| 0 | 203 | 0 | 0 | 2 | 0 | 3 |
| 1 | 203 | 103 | 0 | 2 | 0 | 3 |
| 2 | 203 | 206 | 0 | 2 | 0 | 3 |
| 3 | 203 | 309 | 0 | 2 | 0 | 1 |
| 4 | 203 | 0 | 2 | 1 | 0 | 3 |
| 5 | 203 | 206 | 2 | 1 | 0 | 2 |
| 6 | 203 | 0 | 3 | 1 | 0 | 3 |
| 7 | 203 | 206 | 3 | 1 | 0 | 2 |
| 8 | 203 | 0 | 4 | 1 | 0 | 3 |
| 9 | 203 | 206 | 4 | 1 | 0 | 2 |
| 10 | 1024 | 411 | 0 | 5 | 0 | 6 |
| 11 | 1024 | 617 | 0 | 5 | 0 | 6 |

This corresponds to the following simplified pictorial view:




A note here is that subchannels 6 through 11 are only available when the 7920 is being used. Also, at generation time, the answer to the "TARGET SYSTEM DISC" question can be either 7905 or 7920 for the above technique to work.

Hopefully, this will prove useful.

## RETURNING DAY AND MONTH FROM RTE GREGORIAN DATE

*Larry W. Smith/DSD*

If you were given the Gregorian date (numerical day of the year) and asked to determine the correct day and month of the year, including Leap Year, for purposes of a daily report, how would you do it? The below FORTRAN subroutine illustrates one method:

```
FTN4,L
      SUBROUTINE DATE(IDAY,MONTH,IYEAR)
:C
:C  THIS SUBROUTINE RECEIVES THE GREGORIAN (SOMETIMES
:C  MISTAKINGLY CALLED THE JULIAN) DATE IN 'IDAY'
:C  AND THE YEAR IN 'IYEAR' AND RETURNS THE FOLLOWING:
:C
:C       IDAY ---> DAY OF THE MONTH
:C       MONTH ---> NUMERICAL MONTH OF THE YEAR
:C
      DIMENSION IM(12)
      DATA IM/31,28,31,30,31,30,31,31,30,31,30,31/
:C
:C
:C... CHECK FOR LEAP YEAR ...
:C
      IZ=IYEAR/4
      IR=IYEAR-IZ*4
      IF(IR.NE.0) GO TO 70
:C
:C... LEAP YEAR TIME ...
:C
      IM(2)=29
:C
:C... COMPUTE CORRECT DAY & MONTH ...
:C
   70 DO >? I=1,12
      MONTH=I
     IF(IDAY.LE.IM(I)) GO TO 30
   20 IDAY=IDAY-IM(I)
:C
   30 END

**  NO ERRORS**    PROGRAM = 00090    COMMON = 00000
```

For operating systems which do have access to a normal real-time clock, this could prove to be a handy and inexpensive method. You will also find this subroutine in the LOCUS program 'LUPRN' which lists any RTE device configuration which has recently been updated and modified to include new devices and printing of the time-base select code, privileged interrupt select code, and memory size.

## POWER SUPPLIES ARE IMPORTANT, TOO!!

*Stephen Rutel/DSD*

Hewlett-Packard's new 'B' Version computer products (2109B, 2113B, 12990B, and 12979B) all have something new, the power supply. This new power supply has been designed with service as a prime consideration. On the average, less than a minute is required to completely remove the supply from its mainframe. Slide latches, and new, internal cabling for the supply have been incorporated to make this possible.

Modularity has been utilized inside the supply to make trouble-shooting straightforward and simple. Each board within the supply serves its own individual function, thus allowing problems to be quickly and efficiently tracked down to the source. In this way, service costs are minimized because only boards are replaced, not the complete supply.

The design of the supply incorporates concepts used in its predecessor, the 'A' supply. Switching regulation allows for high power supply efficiency. This means that for power input to the supply, a high percentage (71%) of the power is transferred to the computer mainframe components. This becomes increasingly important as low-line conditions appear. As in the 'A' supply, the 'B' supply has provisions for a power fail recovery system. Should line voltages drop too low for the computer or power supply to function properly (power line failure), semiconductor memory can be maintained for a period of time, dependent on memory size. An added feature of the 'B' supply is a user-accessible connector to allow an external battery to be used in the event of expected extended power failure.

High reliability is also realized in the new supply. The use of state-of-the-art components makes this supply the most dependable yet developed by Data Systems Division. This enables more concentration on computer utilization, and less worry about failures.

How can this power supply benefit you as a Hewlett-Packard 'B' Version computer products owner? High reliability means downtime due to power supply failures are minimized! Serviceability means if the power supply does fail, your system will be back in operation faster! Modularity means parts replacement costs are minimized! Finally, high efficiency means you save electricity without sacrificing performance!

Hewlett-Packard's quality and technical innovations let you compute more and worry less.

# NEW CONTRIBUTED PROGRAMS

*Melanie Van Vliet/DSD*

This article serves as an update for the Data Systems LOCUS Program catalog (22000-90099).

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

## 22682-18963 ENCRYPTION FOR RTE FMGR FILES

This program performs data encryption and decryption in accordance with the Data Encryption Standard (DES) of the National Bureau of Standards, as specified in NBS publication FIPS-PUB-46. The program operates upon FMGR files and will run in RTE, RTE-II, or RTE-III operating systems. Files can also be purged by the program so as to leave no residue of the original data on the disc. Encryption and decryption are performed "in place", using the original storage space to store the encrypted data. Keys may be up to 64 bits long.

| | | |
|---|---|---|
| 22682-18963 | PT | $30.00 |
| 22682-13363 | Cass | $35.00 |

## 22682-18964 RTE SYSTEM DATE & TIME AUTO SCHEDULE

Program times provides an easy method to set the RTE system date and time to automatically schedule programs. Time prints the current date and time and then requests the operator to enter the date and time. If the date and time are correct, the operator merely presses the return key. If they are incorrect, the operator may enter the correct date and time in any of the following formats or combinations thereof. Format examples are for March 10, 1977 2:15:20 P.M.

FORMAT EXAMPLES
3/10/77 14:15:20
Mar 10,1977 14:15:20
March 10,77 14:15,20
3/10/l977,14,15,20
3,10,77,14,15,20
etc.....

Program time will then update the system clock and schedule (place in the time list) any programs contained in a type 4 FMGR file named PSHED. If an absolute starting time is not specified for a program, then it will be scheduled to start at the next resolution code multiple. Subroutine date returns the month, day, year, hour, minute, and second to the calling program. "DATE" calls the system clock and converts it to the following array format.

| WORDS | FORMAT |
|---|---|
| 1&2 | ASCII MONTH |
| 3 | INTEGER DAY |
| 4 | INTEGER YEAR |
| 5 | INTEGER HOUR |
| 6 | INTEGER MINUTE |
| 7 | INTEGER SECOND |

| | | |
|---|---|---|
| 22682-18964 | PT | $10.00 |
| 22682-13364 | Cass | $35.00 |

## 22682-18965 21MX EXTENDED INSTRUCTION SET SIMULATOR

This group of subroutines simulates all bit, byte, word, and index instructions of the 21MX. The calling sequence and results are identical. These routines make the complete line of 21XX computers downward compatible. Total memory required is approximately 500 words. These subroutines are independent of an operating system.

| | | |
|---|---|---|
| 22682-18965 | PT | $25.00 |
| 22682-13365 | Cass | $40.00 |

## 22682-18966 HP-IB PERFORMANCE UTILITY

This performance utility is a series of programs written to characterize the real-time mini-computer on the HP-IB. RTE-II/III/M

| | | |
|---|---|---|
| 22682-10966 | 800 BPI MT | $40.00 |
| 22682-11966 | 1600 BPI MT | $40.00 |
| 22682-13366 | Cass | $35.00 |

## 22682-18967 RTE-II/III FMGR FILE TYPE 2 ASCII, INTEGER, AND REAL SORT PROGRAM PACKAGE

This SORT Package will sort file FMGR Type 2 file of any size and any mix of ASCII, Integer and/or Real fields in mixed

combinations of ascending and descending orders. The SORT program may be run interactively or scheduled programmatically. The destination file may be the source, a new or existing file. If the destination file is new, it will automatically be created. Calculated sort time for a 132 character (66 word) record and 32000 record file in worst case order is less than 45 minutes on a 7900/21MX RTE system. Actual measured sort time for a 20 character record and 2236 record file is 2 minutes and 21 seconds. Typical application is in conjunction with Image/Query. A Fortran program can access the data base and construct a Type 2 file in which each record is a line of a report. The program then schedules SORT and wait and passes the KEY field descriptions. Upon return the program inserts the proper report headers while printing the report from the sorted file. The SORT PACKAGE consists of programs SORT with associated support subroutines and MSORT. SORT extracts the Key fields from the source file, writes the keys to system tracks and schedules MSORT with wait. MSORT performs a disc/memory Sort. When the Keys are in sorted order, SORT reads the record tags from the system tracks and rewrites the source file in sorted order to the destination file.

| | | |
|---|---|---|
| 22682-18967 | PT | $40.00 |
| 22682-13367 | Cass | $70.00 |
| 22682-10967 | 800 BPI MT | $40.00 |
| 22682-11967 | 1600 BPI MT | $40.00 |

## 22682-18968 HP 2644/2645 BCS DRIVER D.05

Driver D.05 communicates with the HP 2644/2645 CRT via the 12966 buffered asynchronous data communication interface card. Communication is either charater or block mode. The terminal can be line or page strapped. The driver (through the use of EQT subunits) provides for control of the keyboard, CRT, cartridge tape units and printer.

| | | |
|---|---|---|
| 22682-10968 | 800 BPI MT | $40.00 |
| 22682-11968 | 1600 BPI MT | $40.00 |
| 22682-18968 | PT | $40.00 |

The following programs have had revisions made to them since the last publication of the catalog:

| | | |
|---|---|---|
| 1. | 22681–***14<br>date code 1718 | RTE ACTIVITY PROFILE GENERATOR |
| 2. | 22682–***50<br>date code 1718 | TRACK — RTE WORKAREA OWNERSHIP |
| 3. | 22682–***31<br>date code 1718 | USER SPOOL POOL FILE ACCESS OR SPOOLING MADE EASIER |
| 4. | 22682–***19<br>date code 1718 | MAPIO — PRINT RTE I/O CONFIGURATION |

The following program has been Withdrawn from the 1977 LOCUS Program catalog and is no longer available:

| | | |
|---|---|---|
| 1. | 22682–***38 | DOS-IIIB INTERACTIVE EDITOR |

# 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 and updates 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 customer manuals.

Update packages are free of charge. If you require an update package only, send your request to:

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

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

A few words about documentation terms:

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

Revised — 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 an update package is incorporated into the manual: the manual gets a new print date and the update package disappears. Note that a revision to a manual effectively obsoletes the previous version of the manual.

Update — An update package is a supplement to an existing manual which contains new and/or changed information. Updates are issued when information must get to customers, yet it is inappropriate to issue a revised manual. An update 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 | DATE | UPDATE |
|---|---|---|---|---|
| 02170-90006 | HP 1000 Computer System Installation and Service | $ 2.50 | 3/77 | 6/77 |
| 02172-90005 | Getting Started with Your HP 1000 | 4.00 | 6/77*R | |
| 91780-93001 | RJE/1000 Programming Manual | 9.50 | 11/76 | 6/77 |

## RTE SYSTEMS MANUALS

| PART NUMBER | MANUAL TITLE | PRICE | DATE | UPDATE |
|---|---|---|---|---|
| 02313-93002 | RTE 2313B Analog-Digital Interface Subsystem Operating and Manual | $30.00 | 8/76 | |
| 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 | 1.00 | 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 |
| 09601-93007 | 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 | |
| 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 | 7/76 | |
| 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 | |

| PART NUMBER | MANUAL TITLE | PRICE | DATE | UPDATE |
|---|---|---|---|---|
| 12732-90001 | RTE Driver DVR33 Programming Manual | $ 2.00 | 2/77*N | |
| 13197-90001 | RTE Driver DVR36 Programming and Operating Manual | 3.00 | 9/76 | |
| 24998-90001 | DOS/RTE Relocatable Library Reference Manual | 10.00 | 5/77*R | |
| 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 Driver DVR62 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*R | |
| 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 System Driver for DVM/Scanner Subsystem | 9.00 | 8/74 | |
| 91700-93001 | Distributed System CCE Operating Manual | 20.00 | 12/76 | |
| 91705-93001 | Distributed System SCE/5 Operating Manual | 15.00 | 12/76 | |
| 91200-90005 | RTE Driver DVA13 for TV Interface (HP 91200B) | 1.50 | 5/77*R | |
| 92001-90015 | RTE DVR05 for 264X Terminals | 2.00 | 9/76 | |
| 92001-93001 | RTE-II Software System Programming and Operating Manual | 10.00 | 7/77*R | |
| 92060-90004 | RTE-III Software System Programming and Operating Manual | 12.00 | 7/77*R | |
| 92060-90005 | RTE Assembler Reference Manual | 7.00 | 12/76 | |
| 92060-90009 | RTE-III General Information Manual | 4.00 | 2/76 | |
| 92060-90010 | RTE Batch/Spool Monitor and Operating System Pocket Guide | 3.00 | 4/77*R | |
| 92060-90012 | RTE: A Guide for New Users | 6.50 | 7/76 | |
| 92060-90013 | Batch-Spool Monitor Reference Manual | 9.50 | 3/77*R | |
| 92060-90014 | RTE Interactive Editor Reference Manual | 6.00 | 3/76 | |
| 92060-90017 | RTE Utility Programs | 3.00 | 3/77*R | |
| 92060-90020 | RTE On-Line Generator | 15.00 | 7/77*R | |
| 92064-90002 | RTE-M Programmer's Reference Manual | 14.00 | 3/77*N | 7/77 |
| 92064-90003 | RTE-M System Generation Reference Manual | 7.50 | 3/77*N | 7/77 |
| 92064-90004 | RTE-M Editor Reference Manual | 6.00 | 1/77 | 3/77 |
| 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 | 7/76 | 3/77 |
| 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 | DATE | UPDATE |
|---|---|---|---|---|
| 02108-90002 | HP 21MX Reference Manual | $ 5.50 | 6/76 | 7/76 |
| 02108-90006 | HP 21MX Installation and Service Manual | 10.00 | 7/76 | |
| 02108-90004 | HP 21MX Operators Manual | 5.00 | 7/76 | |
| 02109-90001 | HP 21MX E-Series Computer Operating and Reference Manual | 8.00 | 9/76 | 3/77 |
| 02109-90002 | HP 21MX E-Series Installation and Service Manual | 15.00 | 8/76 | 3/77 |
| 02109-90006 | HP 21MX M- and E-Series I/O Interfacing Guide | 7.50 | 12/76 | |
| 12979-90007 | HP 12979A I/O Extender Operating and Reference Manual | 5.00 | 12/75 | |
| 12979-90006 | HP 12979A I/O Extender Installation and Service Manual | 15.00 | 6/75 | 12/76 |
| 12990-90003 | HP 12990A Memory Extender Installation and Service Manual | 5.50 | 4/76 | 8/76 |

## LANGUAGE MANUAL

| PART NUMBER | MANUAL TITLE | PRICE | DATE | UPDATE |
|---|---|---|---|---|
| 02100-90140 | Decimal String Arithmetic Routines | $ 6.50 | 2/77*R | |
| 02108-90032 | HP 21MX M-Series Computer RTE Microprogramming Reference Manual | 15.00 | 10/76*R | |
| 02108-90034 | HP 21MX M-Series Computer RTE Microprogramming Pocket Guide | 2.75 | 1/77*N | |
| 02109-90004 | 21MX E-Series 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*R | |
| 12907-90010 | Implementing the HP 2100 Fast FORTRAN Processor | 1.00 | 7/76 | |
| 24307-90014 | DOS-III Assembler Reference Manual | 8.00 | 7/74 | 11/75 |
| 92060-90005 | RTE Assembler Reference Manual | 7.00 | 12/76 | |
| 92060-90016 | Multi-User Real-Time BASIC Reference Manual | 12.00 | 2/77 | 4/77 |
| 92060-90023 | RTE FORTRAN IV Reference Manual | 10.00 | 7/77*N | |
| 92063-90001 | IMAGE/1000 Data Base Management System Reference Manual | 9.00 | 2/77*R | 6/77 |
| 92065-90001 | RTE-M Real-Time BASIC Language Reference Manual | 8.50 | 2/77 | |

# SOFTWARE UPDATES

Following are cross-reference lists of the available 92001B, 92060B, 92062A, 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 1726. Software modules updated since the last issue are indicated for easy reference.

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 1726 (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).

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | DATE CODE |
|---|---|---|---|---|
| 02607-16004 | !S4L07 | 24K SIO LINE PRINTER DRIVER | 92001-13305 | 1538 |
| 09601-16021 | %DVR15 | RTE 7261A DRIVER | 92062-13304 | A |
| 12732-16001 | %DVR33 | FLEXIBLE DISC DRIVER | 92062-13304 | 1726* |
| 12970-16004 | !S4MT1 | 24K SIO MAG. TAPE DRIVER | 92001-13305 | 1550 |
| 20747-60001 | %DVR30 | RTE FIXED HEAD DISC DRIVER | 92062-13305 | C |
| 20808-60001 | %CAL10 | CAL. PLOTTER DRIVER | 92062-13302 | B |
| 20810-60001 | %CALIB | CAL. PLOTTER LIBRARY | 92062-13302 | C |
| 20875-60001 | %1FTN | FORTRAN MAIN CONTROL | 92060-13308 | E |
| 20875-60002 | %2FTN | FORTRAN PASS 1 | 92060-13308 | E |
| 20875-60003 | %3FTN | FORTRAN PASS 2 | 92060-13308 | E |
| 20875-60004 | %4FTN | FORTRAN PASS 3 | 92060-13308 | F |
| 20875-60005 | %5FTN | FORTRAN PASS 4 | 92060-13308 | F |
| 24129-60001 | %ALGOL | RTE/DOS ALGOL PART 1 | 92060-13305 | 1643 |
| 24129-60002 | %ALGL1 | RTE/DOS ALGOL PART 2 | 92060-13305 | C |
| 24153-60001 | %FF.N | RTE/DOS FORMATTER | 92060-13303 | C |
| 24306-60001 | %DECAR | DOSM ST ARITH PK | 92060-13303 | A |
| 24998-16001 | %RLIB1 | RTE/DOS LIBRARY PART 1 | 92060-13302 | 1726* |
| 24998-16001 | %RLIB2 | RTE/DOS LIBRARY PART 2 | 92060-13302 | 1726* |
| 24998-16002 | %FF4.N | FORTRAN IV FORMATTER | 92060-13303 | 1726* |
| 25117-60499 | %DVR24 | RTE 7970 7T MAG. TAPE DRIVER | 92062-13305 | D |
| 29013-60001 | %DVR31 | RTE 7900A DISC DRIVER | 92062-13305 | 1710 |
| 29028-60002 | %DVR12 | RTE 2767A DRIVER | 92062-13303 | A |
| 29029-60001 | %DVR00 | RTE TTY/PUNCH/PHOTO READER | 92062-13302 | 1642 |
| 29030-60001 | %DVR11 | RTE 2892A CARD READER DRIVER | 92062-13303 | 1710 |
| 29100-60017 | !S4LP | 24K SIO LINE PRINTER | 92001-13305 | A |
| 29100-60018 | !S4SYD | 24K SIO SYSTEM DUMP | 92001-13305 | A |
| 29100-60019 | !S4PHR | 24K SIO PHOTO READER | 92001-13305 | A |
| 29100-60020 | !S4PUN | 24K SIO TAPE PUNCH | 92001-13305 | A |
| 29100-60022 | !S4L67 | 24K SIO 2767 LINE PRINTER | 92001-13305 | A |
| 29100-60023 | !S4MT7 | 24K SIO 7970 MAG.TAPE | 92001-13305 | A |
| 29100-60049 | !S4MT3 | 24K SIO MAG. TAPE | 92001-13305 | A |
| 29100-60050 | !S4TER | 24K SIO TERMINAL PRINTER | 92001-13305 | A |
| 59310-16002 | %1DV37 | RTE HP-IB WITHOUT SRQ | 92062-13304 | 1726* |
| 59310-16003 | %2DV37 | RTE HP-IB WITH SRQ | 92062-13304 | 1726* |
| 59310-16004 | %HPIR | HP-IB DEVICE SUBROUTINE | 92062-13304 | 1710 |
| 59310-16005 | %SRQ.P | SRQ.P TRAP UTILITY | 92062-13304 | 1710 |
| 72008-60001 | %1DV10 | COMP. 7210A PLOTTER DRIVER | 92062-13302 | A |
| 72009-60001 | %2DV10 | MIN. 7210A PLOTTER DRIVER | 92062-13302 | A |
| 91200-16001 | %DVA13 | 91200A DRIVER | 92062-13303 | 1648 |
| 91200-16002 | %TVLIB | 91200A VIDEO MONITOR LIBRARY | 92062-13303 | 1648 |
| 91200-16004 | %TVVER | 91200A TV INTERFACE VERIFIER | 92062-13303 | 1648 |
| 92001-16002 | %LDR2 | RTE LOADER | 92001-13301 | 1726* |
| 92001-16003 | %MTM | MULT. TERMINAL MONITOR | 92001-13301 | H |
| 92001-16004 | %2DP43 | POWER FAILURE DRIVER | 92001-13301 | 1633 |
| 92001-16005 | %SYLIB | RTE SYSTEM LIBRARY | 92001-13301 | 1726* |
| 92001-16012 | %CR2SY | CORE RESIDENT OPERATING SYS. | 92001-13301 | 1726* |

(Continued)

SOFTWARE MODULE NUMBERS: 92001B LEVEL 1726 (RTEII)

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | DATE CODE |
|---|---|---|---|---|
| 92001-16013 | !2GN00 | RTE-II 7900 OFF-LINE GEN. | 92001-13303 | 1631 |
| 92001-16014 | %AUTOP | AUTO RESTART PROGRAM | 92001-13302 | 1631 |
| 92001-16018 | !2GNFH | RTE-II FIXED HEAD DISC GEN. | 92001-13306 | 1631 |
| 92001-16020 | %DVA12 | 2607/10/13/14/17/18 DRIVER | 92062-13303 | 1534 |
| 92001-16026 | !2GN05 | RTE-II 7905 OFF-LINE GEN. | 92001-13303 | 1631 |
| 92001-16027 | %4DV05 | RTE 2644/45 DRIVER | 92062-13302 | 1650 |
| 92001-16028 | %0DV05 | RTE 2640A DRIVER | 92062-13302 | 1650 |
| 92001-16029 | %%CMD2 | RTE-II COMMAND PROGRAM | 92001-13301 | 1710 |
| 92001-16030 | %WHZAT | RTE-II WHZAT PROGRAM | 92001-13302 | 1726* |
| 92001-16031 | %RT2G1 | RTE-II ON-LINE GENERATOR PT. 1 | 92001-13304 | 1704 |
| 92001-16031 | %RT2G2 | RTE-II ON-LINE GENERATOR PT. 1 | 92001-13304 | 1704 |
| 92001-18014 | &AUTOR | AUTO RESTART SOURCE | 92001-13302 | 1631 |
| 92001-18033 | &AN2F0 | RTE-II 7900 GFATHER ANSW FILE | 92001-13307 | 1631 |
| 92001-18034 | &AN2F5 | RTE-II 7905 GFATHER ANSW FILE | 92001-13307 | 1631 |
| 92002-12001 | %BMPG1 | BATCH MONITOR PROGRAM PART 1 | 92002-13301 | 1631 |
| 92002-12001 | %BMPG2 | BATCH MONITOR PROGRAM PART 2 | 92002-13301 | 1631 |
| 92002-12001 | %BMPG3 | BATCH MONITOR PROGRAM PART 3 | 92002-13301 | 1631 |
| 92002-12002 | %2SP01 | RTE-II SPOOL MONITOR PART 1 | 92002-13303 | 1631 |
| 92002-12002 | %2SP02 | RTE-II SPOOL MONITOR PART 2 | 92002-13303 | 1631 |
| 92002-16006 | %BMLIB | BATCH LIBRARY | 92002-13302 | 1631 |
| 92002-16010 | %EDITR | RTE EDITOR | 92002-13302 | C |
| 92060-12004 | %ASMB | RTE ASSEMBLER | 92060-13304 | 1634 |
| 92060-12005 | %CLIB | RTE COMPILER LIBRARY | 92060-13315 | 1726* |
| 92060-16028 | %XREF | CROSS REFERENCE | 92060-13304 | A |
| 92060-16031 | %DVR32 | RTE 7905A DISC DRIVER | 92062-13305 | A |
| 92060-16038 | %SWTCH | RTE-II SWITCH PROGRAM | 92001-13304 | 1710 |
| 92060-16039 | %SAVE | SAVE PROGRAM | 92060-13309 | 1704 |
| 92060-16040 | %RESTR | RESTORE PROGRAM | 92060-13309 | 1704 |
| 92060-16041 | %VERFY | DISC VERIFY PROGRAM | 92060-13309 | 1704 |
| 92060-16042 | %COPY | DISC COPY PROGRAM | 92060-13309 | 1704 |
| 92060-16043 | %DBKLB | DISC BACK UP LIBRARY | 92060-13309 | 1704 |
| 92060-16044 | !DSKUP | OFF LINE DISC BACK UP | 92060-13309 | 1704 |
| 92060-16045 | %RDNAM | READ NAME PROGRAM | 92001-13302 | 1631 |
| 92060-16052 | %KEYS | SOFT KEY UTILITY | 92001-13002 | 1707 |
| 92060-16053 | %KYDMP | SOFT KEY DUMP UTILITY | 92001-13002 | 1707 |
| 92060-16092 | %FTN4 | RTE FORTRAN IV MAIN | 92060-13316 | 1726* |
| 92060-16093 | %FFTN4 | RTE FORTRAN IV SEG F | 92060-13316 | 1726* |
| 92060-16094 | %0FTN4 | RTE FORTRAN IV SEG 0 | 92060-13316 | 1726* |
| 92060-16095 | %1FTN4 | RTE FORTRAN IV SEG 1 | 92060-13316 | 1726* |
| 92060-16096 | %2FTN4 | RTE FORTRAN IV SEG 2 | 92060-13316 | 1726* |
| 92060-16097 | %3FTN4 | RTE FORTRAN IV SEG 3 | 92060-13316 | 1726* |
| 92060-16098 | %4FTN4 | RTE FORTRAN IV SEG 4 | 92060-13316 | 1726* |
| 92060-18046 | &UPDAT | UPDATE TRANSFER FILE | 92001-13302 | 1631 |
| 92060-18047 | &PKDIS | PACK DISC TRANSFER FILE | 92001-13302 | 1631 |
| 92202-16001 | %DVR23 | RTE 7970 9T. MAG. TAPE DRIVER | 92062-13304 | A |
| 92900-16002 | %2DV47 | RTE 92900A DRIVER WITHOUT DMS | 92062-13302 | 1643 |

## SOFTWARE MODULE NUMBERS: 92060B LEVEL 1726 (RTE-III)

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

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | DATE CODE |
|---|---|---|---|---|
| 02607-16004 | !S4L07 | 24K STD LINE PRINTER DRIVER | 92001-13305 | 1538 |
| 09601-16021 | %DVR15 | RTE 7261A DRIVER | 92062-13304 | A |
| 12732-16001 | %DVR33 | FLEXIBLE DISC DRIVER | 92062-13304 | 1726* |
| 12970-16004 | !S4MT1 | 24K STD MAG.TAPE DRIVER | 92001-13305 | 1550 |
| 20747-60001 | %DVR30 | RTE FIXED HEAD DISC DRIVER | 92062-13305 | C |
| 20808-60001 | %CAL10 | CAL. PLOTTER DRIVER | 92062-13302 | B |
| 20810-60001 | %CALIB | CAL. PLOTTER LIBRARY | 92062-13302 | C |
| 20875-60001 | %1FTN | FORTRAN MAIN CONTROL | 92060-13308 | E |
| 20875-60002 | %2FTN | FORTRAN PASS 1 | 92060-13308 | E |
| 20875-60003 | %3FTN | FORTRAN PASS 2 | 92060-13308 | E |
| 20875-60004 | %4FTN | FORTRAN PASS 3 | 92060-13308 | E |
| 20875-60005 | %5FTN | FORTRAN PASS 4 | 92060-13308 | E |
| 24129-60001 | %ALGOL | RTE/DOS ALGOL PART 1 | 92060-13305 | 1643 |
| 24129-60002 | %ALGL2 | RTE/DOS ALGOL PART 2 | 92060-13305 | C |
| 24153-60001 | %FF.N | RTE/DOS FORMATTER | 92060-13303 | C |
| 24306-60001 | %DFCAR | DOSM ST ARITH PK | 92060-13303 | A |
| 24998-16001 | %RLIB1 | RTE/DOS LIBRARY PART 1 | 92060-13302 | 1726* |
| 24998-16001 | %RLIB2 | RTE/DOS LIBRARY PART 2 | 92060-13302 | 1726* |
| 24998-16002 | %FF4.N | FORTRAN IV FORMATTER | 92060-13303 | 1726* |
| 25117-60499 | %DVR24 | RTE 7970 7T MAG. TAPE DRIVER | 92062-13305 | D |
| 29013-60001 | %DVR31 | RTE 7900A DISC DRIVER | 92062-13305 | 1710 |
| 29028-60002 | %DVR12 | RTE 2767A DRIVER | 92062-13303 | A |
| 29029-60001 | %DVR00 | RTE TTY/PUNCH/PHOTO READER | 92062-13302 | 1642 |
| 29030-60001 | %DVR11 | RTE 2892A CARD READER DRIVER | 92062-13303 | 1710 |
| 29100-60017 | !S4LP | 24K STD LINE PRINTER | 92001-13305 | A |
| 29100-60018 | !S4SYD | 24K STD SYSTEM DUMP | 92001-13305 | A |
| 29100-60019 | !S4PHP | 24K STD PHOTO READER | 92001-13305 | A |
| 29100-60020 | !S4PUN | 24K STD TAPE PUNCH | 92001-13305 | A |
| 29100-60022 | !S4L67 | 24K STD 2767 LINE PRINTER | 92001-13305 | A |
| 29100-60023 | !S4MT2 | 24K STD 7970 MAG. TAPE | 92001-13305 | A |
| 29100-60049 | !S4MT3 | 24K STD MAG. TAPE | 92001-13305 | A |
| 29100-60050 | !S4TER | 24K STD TERMINAL PRINTER | 92001-13305 | A |
| 59310-16002 | %1DV37 | RTE HP-IB WITHOUT SRQ | 92062-13304 | 1726* |
| 59310-16003 | %2DV37 | RTE HP-IB WITH SRQ | 92062-13304 | 1726* |
| 59310-16004 | %HPIB | HP-IB DEVICE SUBROUTINE | 92062-13304 | 1710 |
| 59310-16005 | %SRQ.P | SRQ.P TRAP UTILITY | 92062-13304 | 1710 |
| 72008-60001 | %1DV10 | COMP. 7210A PLOTTER DRIVER | 92062-13302 | A |
| 72009-60001 | %2DV10 | MIN. 7210A PLOTTER DRIVER | 92062-13302 | A |
| 91200-16001 | %DVA13 | 91200A DRIVER | 92062-13303 | 1648 |
| 91200-16002 | %TVLIB | 91200A VIDEO MONITOR LIBRARY | 92062-13303 | 1648 |
| 91200-16004 | %TVVER | 91200A TV INTERFACE VERIFIER | 92062-13303 | 1648 |
| 92001-16003 | %MTM | MULT. TERMINAL MONITOR | 92060-13301 | H |
| 92001-16005 | %SYLIB | RTE SYSTEM LIBRARY | 92060-13301 | 1726* |
| 92001-16014 | %AUTOR | AUTO RESTART PROGRAM | 92060-13310 | 1631 |
| 92001-16020 | %DVA12 | 2607/10/13/14/17/18 DRIVER | 92062-13303 | 1534 |
| 92001-16027 | %4DV05 | RTE 2644/45 DRIVER | 92062-13302 | 1650 |

(Continued)

SOFTWARE MODULE NUMBERS: 92060B LEVEL 1726 (RTE III)

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | DATE CODE |
|---|---|---|---|---|
| 92001-16028 | %0DV05 | RTE 2640A DRIVER | 92062-13302 | 1650 |
| 92001-18014 | &AUTOR | AUTO RESTART PROGRAM SOURCE | 92060-13310 | 1631 |
| 92002-12001 | %BMPG1 | BATCH MONITOR PROGRAM PART 1 | 92002-13301 | 1631 |
| 92002-12001 | %BMPG2 | BATCH MONITOR PROGRAM PART 2 | 92002-13301 | 1631 |
| 92002-12001 | %BMPG3 | BATCH MONITOR PROGRAM PART 3 | 92002-13301 | 1631 |
| 92002-16006 | %BMLIB | BATCH LIBRARY | 92002-13302 | 1631 |
| 92002-16010 | %EDITR | RTE EDITOR | 92002-13302 | C |
| 92060-12001 | %3SPO1 | RTE-III SPOOL MONITOR PART 1 | 92060-13313 | 1631 |
| 92060-12001 | %3SPO2 | RTE-III SPOOL MONITOR PART 2 | 92060-13313 | 1631 |
| 92060-12003 | %CR3SY | MEMORY RESIDENT SYSTEM | 92060-13301 | 1726* |
| 92060-12004 | %ASMB | RTE ASSEMBLER | 92060-13304 | 1639 |
| 92060-12005 | %CLIB | RTE COMPILER LIBRARY | 92060-13315 | 1726* |
| 92060-16001 | %3DP43 | POWER FAILURE DRIVER | 92060-13301 | 1633 |
| 92060-16004 | %LDR3 | RTE-III LOADER | 92060-13301 | 1726* |
| 92060-16006 | %WHZT3 | RTE-III WHZAT PROGRAM | 92060-13310 | 1726* |
| 92060-16028 | %XREF | CROSS REFERENCE | 92060-13304 | A |
| 92060-16029 | !3GN00 | 7900 RTE-III GENERATOR | 92060-13311 | 1631 |
| 92060-16031 | %DVR32 | RTE 7905A DISC DRIVER | 92062-13305 | A |
| 92060-16032 | !3GN05 | 7905 RTE-III GENERATOR | 92060-13311 | 1631 |
| 92060-16035 | %$PVMP | $PVMP | 92060-13301 | A |
| 92060-16036 | %$CMD3 | RTE-III COMMAND PROGRAM | 92060-13301 | 1710 |
| 92060-16037 | %RT3G1 | RTE-III ON-LINE GENERATOR PT.1 | 92060-13312 | 1704 |
| 92060-16037 | %RT3G2 | RTE-III ON-LINE GENERATOR PT.2 | 92060-13312 | 1704 |
| 92060-16038 | %SWTCH | RTE-III SWITCH PROGRAM | 92060-13312 | 1710 |
| 92060-16039 | %SAVE | SAVE PROGRAM | 92060-13309 | 1704 |
| 92060-16040 | %RESTR | RESTORE PROGRAM (PSTOR) | 92060-13309 | 1704 |
| 92060-16041 | %VERFY | DISC VERIFY PROGRAM | 92060-13309 | 1704 |
| 92060-16042 | %COPY | DISC COPY PROGRAM | 92060-13309 | 1704 |
| 92060-16043 | %DMKLB | DISK PACK UP LIBRARY | 92060-13309 | 1704 |
| 92060-16044 | !DSKUP | OFF LINE DISK BACK UP | 92060-13309 | 1704 |
| 92060-16045 | %RDNAM | READ NAME PROGRAM | 92060-13310 | 1631 |
| 92060-16052 | %KEYS | SOFT KEY UTILITY | 92062-13310 | 1707 |
| 92060-16053 | %KYDMP | SOFT KEY DUMP UTILITY | 92060-13310 | 1707 |
| 92060-16092 | %FTN4 | RTE FORTRAN IV MAIN | 92060-13316 | 1726* |
| 92060-16093 | %FFTN4 | FORTRAN IV SEGMENT F | 92060-13316 | 1726* |
| 92060-16094 | %0FTN4 | FORTRAN IV SEGMENT 0 | 92060-13316 | 1726* |
| 92060-16095 | %1FTN4 | FORTRAN IV SEGMENT 1 | 92060-13316 | 1726* |
| 92060-16096 | %2FTN4 | FORTRAN IV SEGMENT 2 | 92060-13316 | 1726* |
| 92060-16097 | %3FTN4 | FORTRAN IV SEGMENT 3 | 92069-13316 | 1726* |
| 92060-16098 | %4FTN4 | FORTRAN IV SEGMENT 4 | 92060-13316 | 1726* |
| 92060-18046 | &UPDAT | UPDATE TRANSFER FILE | 92060-13310 | 1631 |
| 92060-18047 | &PKDIS | PACK DISK TRANSFER FILE | 92060-13310 | 1631 |
| 92060-18050 | &AN3F0 | RTE-III 7900 GFATHER ANSW FILE | 92060-13314 | 1726* |
| 92060-18051 | &AN3F5 | RTE-III 05/20 GFATHER ANS FILE | 92060-13314 | 1726* |
| 92202-16001 | %DVR23 | RTE 7970 9T. MAG. TAPE DRIVER | 92062-13304 | A |
| 92900-16002 | %2DV47 | RTE 92900A DRIVER WITHOUT DMS | 92062-13302 | 1726* |

## SOFTWARE MODULE NUMBERS: 92062B LEVEL 1726 (RTE-III)

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | DATE CODE |
|---|---|---|---|---|
| 09601-16021 | %DVR15 | RTE 7261A DRIVER | 92062-13304 | A |
| 12732-16001 | %DVR33 | FLEXIBLE DISC DRIVER | 92062-13304 | 1726* |
| 20747-60001 | %DVR30 | RTE FIXED HEAD DISC DRIVER | 92062-13305 | C |
| 20808-60001 | %CAL10 | CAL. PLOTTER DRIVER | 92062-13302 | B |
| 20810-60001 | %CALIB | CAL. PLOTTER LIBRARY | 92062-13302 | C |
| 25117-60499 | %DVR24 | RTE 7970 7T MAG. TAPE DRIVER | 92062-13305 | D |
| 29013-60001 | %DVR31 | RTE 7900A DISC DRIVER | 92062-13305 | 1710 |
| 29028-60002 | %DVR12 | RTE 2767A DRIVER | 92062-13303 | A |
| 29029-60001 | %DVR00 | RTE TTY/PUNCH/PHOTO READER | 92062-13302 | 1642 |
| 29030-60001 | %DVR11 | RTE 2892A CARD READER DRIVER | 92062-13303 | 1710 |
| 59310-16002 | %1DV37 | RTE HP-IB WITHOUT SRQ | 92062-13304 | 1726* |
| 59310-16003 | %2DV37 | RTE HP-IB WITH SRQ | 92062-13304 | 1726* |
| 59310-16004 | %HPIB | HP-IB DEVICE SUBROUTINE | 92062-13304 | 1710 |
| 59310-16005 | %SRQ.P | SRQ.P TRAP UTILITY | 92062-13304 | 1710 |
| 72008-60001 | %1DV10 | COMP. 7210A PLOTTER DRIVER | 92062-13302 | A |
| 72009-60001 | %2DV10 | MIN. COMP. 7910A PLOTER DRIVE | 92062-13302 | A |
| 91200-16001 | %DVA13 | 91200A DRIVER | 92062-13303 | 1648 |
| 91200-16002 | %TVLIB | 91200A VIDEO MONITOR LIBRARY | 92062-13303 | 1648 |
| 91200-16004 | %TVVFR | 91200A TV INTERFACE VERIFIER | 92062-13303 | 1648 |
| 92001-16020 | %DVA12 | 2607/10/13/14/17/18 DRIVER | 92062-13303 | 1534 |
| 92001-16027 | %4DV05 | RTE 2644/45 DRIVER | 92062-13302 | 1650 |
| 92001-16028 | %0DV05 | RTE 2640A DRIVER | 92062-13302 | 1650 |
| 92060-16031 | %DVR32 | RTE 7905A DISC DRIVER | 92062-13305 | 1704 |
| 92202-16001 | %DVR23 | RTE 7970 9T. MAG. TAPE DRIVER | 92062-13304 | A |
| 92900-16002 | %2DV47 | RTE 92900A DRIVER WITHOUT DMS | 92062-13302 | 1643 |
| 92900-16003 | %3DV47 | RTE 92900A DRIVER WITH DMS | 92062-13302 | 1643 |

## SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1726 (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 (MIII) 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 1726 (RTE-M)

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | FLEXIBLE DISC | DATE CODE |
|---|---|---|---|---|---|
| 09601-16021 | %DVR15 | RTE 7261A CARD READER DRIVER | 92062-13304 | 92064-13401 | A |
| 12732-16001 | %DVR33 | FLEXIBLE DISC DRIVER | 92062-13304 | 92064-13401 | 1650 |
| 20808-60001 | %CAL10 | RTE PLOTTER DRIVER | 92062-13302 | 92064-13401 | B |
| 20810-60001 | %CALIB | CAL. PLOTTER LIBRARY | 92062-13302 | 92064-13401 | C |
| 24153-60001 | %FF.N | RTE/DOS FORTRAN FORMATTER | 92060-13303 | 92064-13401 | C |
| 24153-60001 | %FF.N | RTE/DOS FORTRAN FORMATTER | 92060-13303 | 92064-13402 | C |
| 24306-60001 | %DECAR | DOSM STRING ARITH PK | 92060-13303 | | A |
| 24998-16001 | %RLIB1 | RTE/DOS LIBRARY | 92060-13302 | 92064-13402 | 1624 |
| 24998-16001 | %RLIB1 | RTE/DOS LIBRARY | 92060-13302 | 92064-13401 | 1624 |
| 24998-16001 | %RLIB2 | RTE/DOS LIBRARY | 92060-13302 | 92064-13401 | 1624 |
| 24998-16001 | %RLIB2 | RTE/DOS LIBRARY | 92060-13302 | 92064-13402 | 1624 |
| 24998-16002 | %FF4.N | FORTRAN IV FORMATTER | 92060-13303 | 92064-13402 | 1624 |
| 24998-16002 | %FF4.N | FORTRAN IV FORMATTER | 92060-13303 | 92064-13401 | 1624 |
| 29028-60002 | %DVR12 | RTE 2767A DRIVER | 92062-13303 | 92064-13401 | A |
| 29029-60001 | %DVR00 | RTE TTY/PUNCH/PHOTO READER | 92062-13302 | 92064-13401 | 1642 |
| 29030-60001 | %DVR11 | RTE 2892A CARD READER DRIVER | 92062-13303 | 92064-13401 | 1710 |
| 59310-16002 | %1DV37 | HP-IB WITHOUT SYSTEM REQUEST | 92062-13304 | 92064-13401 | 1710 |
| 59310-16003 | %2DV37 | HP-IB WITH SYSTEM REQUEST | 92062-13304 | 92064-13401 | 1710 |
| 59310-16004 | %HPIB | HP-IB RTE UTILITY | 92062-13304 | 92064-13401 | 1710 |
| 59310-16005 | %SRQ.P | SRQ.P TRAP UTILITY | 92062-13304 | 92064-13401 | 1710 |
| 72008-60001 | %1DV10 | COMP. 7210A PLOTTER DRIVER | 92062-13302 | 92064-13401 | A |
| 72009-60001 | %2DV10 | MIN. COMP. 7210A PLOTTER DRIVE | 92062-13302 | 92064-13401 | A |
| 91200-16001 | %DVA13 | 91200 TV INTERFACE DRIVER | 92062-13303 | 92064-13401 | 1648 |
| 91200-16002 | %TVLIB | VIDEO MONITOR LIBRARY | 92062-13303 | 92064-13401 | 1648 |
| 91200-16004 | %TVVER | TV INFT VERIF | 92062-13303 | 92064-13401 | 1648 |
| 92001-16020 | %DVA12 | 2607/10/13/14/17/18 DRIVER | 92062-13303 | 92064-13401 | 1534 |
| 92001-16027 | %4DV05 | RTE 2644/45 DRIVER | 92062-13302 | 92064-13401 | 1650 |
| 92001-16028 | %0DV05 | RTE 2640A DRIVER | 92062-13302 | 92064-13401 | 1650 |
| 92060-16052 | %KEYS | SOFT KEY UTILITY | 92064-13304 | 92064-13402 | 1707 |
| 92060-16053 | %KYDMP | SOFT KEY DUMP UTILITY | 92064-13304 | 92064-13402 | 1707 |
| 92060-16092 | %FTN4 | FORTRAN IV MAIN | | 92064-13402 | 1726* |
| 92060-16093 | %FFTN4 | RTE FORTRAN IV SEG ID SUB | | 92064-13402 | 1726* |
| 92060-16094 | %0FTN4 | FORTRAN IV SEGMENT 0 | | 92064-13402 | 1726* |
| 92060-16095 | %1FTN4 | FORTRAN IV SEGMENT 1 | | 92064-13402 | 1726* |
| 92060-16096 | %2FTN4 | FORTRAN IV SEGMENT 2 | | 92064-13402 | 1726* |
| 92060-16097 | %3FTN4 | FORTRAN IV SEGMENT 3 | | 92064-13402 | 1726* |
| 92060-16098 | %4FTN4 | FORTRAN IV SEGMENT 4 | | 92064-13402 | 1726* |
| 92064-12005 | %FMPC | CARTRIDGE FMP/FMPCR (LIB) | 92064-13306 | 92064-13401 | 1709 |
| 92064-12006 | %FMPF | FLEX DISC FMGR LIB (GEN DISC) | | 92064-13401 | 1726* |
| 92064-12006 | %FMPF | FLEX DISC FMGR LIB (APP DISC) | | 92064-13402 | 1726* |
| 92064-12007 | %CLIBM | RTE COMPILER LIBRARY | | 92064-13402 | 1726* |
| 92064-16001 | %MSY1 | MI OPERATING SYSTEM | 92064-13301 | 92064-13401 | 1726* |
| 92064-16002 | %MSY2 | MII OPERATING SYSTEM | 92064-13302 | 92064-13401 | 1726* |
| 92064-16003 | %MSY3 | MIII OPERATING SYSTEM | 92064-13303 | 92064-13401 | 1726* |
| 92064-16005 | %MBU | MI BUFFERING | 92064-13301 | 92064-13401 | 1650 |
| 92064-16006 | %MMP | MI SCHEDULING OPTION | 92064-13301 | 92064-13401 | 1650 |

(Continued)

SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1726 (RTE-M)

| PAPER TAPE | MODULE | DESCRIPTION | CARTRIDGE | FLEXIBLE DISC | DATE CODE |
|---|---|---|---|---|---|
| 92064-16008 | %MTI | TIMER OPTION (MI) | 92064-13301 | 92064-13401 | 1650 |
| 92064-16008 | %MTI | TIMER OPTION (MIII) | 92064-13303 | 92064-13401 | 1650 |
| 92064-16008 | %MTI | TIMER OPTION (MII) | 92064-13302 | 92064-13401 | 1650 |
| 92064-16009 | %MTS | TIME SCHEDULING OPTION (MIII) | 92064-13303 | 92064-13401 | 1650 |
| 92064-16009 | %MTS | TIME SCHEDULING OPTION (MI) | 92064-13301 | 92064-13401 | 1650 |
| 92064-16009 | %MTS | TIME SCHEDULING OPTION (MII) | 92064-13302 | 92064-13401 | 1650 |
| 92064-16010 | %MOP | OPERATOR COMMAND OPTION (MIII) | 92064-13303 | 92064-13401 | 1650 |
| 92064-16010 | %MOP | OPERATOR COMMAND OPTION (MI) | 92064-13301 | 92064-13401 | 1650 |
| 92064-16010 | %MOP | OPERATOR COMMAND OPTION (MII) | 92064-13302 | 92064-13401 | 1650 |
| 92064-16011 | %MCL | CLASS I/O OPTION (MII) | 92064-13302 | 92064-13401 | 1726* |
| 92064-16012 | %MAP | MI/II ABSOLUTE PROGRAM LOADER | 92064-13305 | 92064-13401 | 1726* |
| 92064-16013 | %MDMLB | DUMMY LIBRARY (MII) | 92064-13302 | 92064-13401 | 1650 |
| 92064-16013 | %MDMLB | DUMMY LIBRARY (MIII) | 92064-13303 | 92064-13401 | 1650 |
| 92064-16013 | %MDMLB | DUMMY LIBRARY (MI) | 92064-13301 | 92064-13401 | 1650 |
| 92064-16015 | %MCL3 | CLASS I/O OPTION (MIII) | 92064-13303 | 92064-13401 | 1726* |
| 92064-16016 | %MAP3 | MIII ABSOLUTE PROGRAM LOADER | 92064-13305 | 92064-13401 | 1726* |
| 92064-16017 | %FMGC0 | CARTRIDGE FILE MANAGER | 92064-13305 | 92064-13401 | 1709 |
| 92064-16018 | %DRC | CARTRIDGE DIR MAN PROGRAM | 92064-13304 | 92064-13401 | 1650 |
| 92064-16019 | %TBLCR | CARTRIDGE DIRECTORY TABLES | 92064-13304 | 92064-13401 | 1650 |
| 92064-16021 | %DRC1 | MI CARTRIDGE DIRECTORY SUBR | 92064-13306 | 92064-13401 | 1650 |
| 92064-16022 | %RTMGN | SYSTEM GENERATOR | 92064-13305 | 92064-13401 | 1726* |
| 92064-16023 | %RTMLD | RELOCATING LOADER (APP DISC) | 92064-13305 | 92064-13402 | 1726* |
| 92064-16023 | %RTMLD | RELOCATING LOADER (GEN DISC) | 92064-13305 | 92064-13401 | 1726* |
| 92064-16024 | %RTMSC | LOADER SUB CONTROL (GEN DISC) | 92064-13305 | 92064-13401 | 1726* |
| 92064-16024 | %RTMSC | LOADER SUB CONTROL (APP DISC) | 92064-13305 | 92064-13402 | 1726* |
| 92064-16025 | %MEDIT | EDITOR | | 92064-13402 | 1703 |
| 92064-16026 | %MAS46 | CROSS REFERENCE SEGMENT | | 92064-13402 | 1650 |
| 92064-16027 | %MPF | MI/II POWER FAIL | 92064-13304 | 92064-13401 | 1650 |
| 92064-16029 | %MPF3 | MIII POWER FAIL | 92064-13304 | 92064-13401 | 1650 |
| 92064-16030 | %MAUTO | AUTOR REL | 92064-13304 | 92064-13401 | 1650 |
| 92064-16031 | %MRN | RESOURCE NUMBER MANAGER (MII) | 92064-13302 | 92064-13401 | 1650 |
| 92064-16031 | %MRN | RESOURCE NUMBER MNGR (MIII) | 92064-13303 | 92064-13401 | 1650 |
| 92064-16032 | %ONMTM | MULTI TERMINAL MONITOR (GEN D) | 92064-13305 | 92064-13401 | 1650 |
| 92064-16032 | %ONMTM | MULTI TERMINAL MONITOR (APP D) | 92064-13305 | 92064-13402 | 1650 |
| 92064-16033 | !MCGEN | ABSOLUTE CARTRIDGE GENERATOR | 92064-13307 | | 1726* |
| 92064-16034 | %SGPRP | SEGMENT PROGRAM PREP | | 92064-13402 | 1650 |
| 92064-16035 | %MPRMP | PROMPT (MTM) | 92064-13305 | 92064-13401 | 1650 |
| 92064-16036 | %MRSPN | RESPONSE (MTM) | 92064-13305 | 92064-13401 | 1650 |
| 92064-16040 | %MASM0 | ASSEMBLER MAIN CONTROL | | 92064-13402 | 1650 |
| 92064-16041 | %MASM1 | ASSEMBLER SEGMENT 1 | | 92064-13402 | 1650 |
| 92064-16042 | %MASM2 | ASSEMBLER SEGMENT 2 | | 92064-13402 | 1650 |
| 92064-16043 | %MASM3 | ASSEMBLER SEGMENT 3 | | 92064-13402 | 1650 |
| 92064-16044 | %MASM4 | ASSEMBLER SEGMENT 4 | | 92064-13402 | 1650 |
| 92064-16045 | %MFTN0 | FORTRAN MAIN CONTROL | | 92064-13402 | 1650 |
| 92064-16046 | %MFTN1 | FORTRAN SEGMENT 1 | | 92064-13402 | 1650 |
| 92064-16047 | %MFNT2 | FORTRAN SEGMENT 2 | | 92064-13402 | 1650 |
| 92064-16050 | %MASM5 | ASSEMBLER SEGMENT 0 | | 92064-13402 | 1650 |
| 92064-16051 | %MXRF0 | CROSS REFERENCE MAIN | | 92064-13402 | 1650 |
| 92064-16054 | %DIRD | CARTRIDGE DIRECTORY READ | 92064-13304 | 92064-13401 | 1650 |
| 92064-16055 | %FMGF0 | FLEX DISC FILE MNGR (GEN DISC) | | 92064-13401 | 1709 |
| 92064-16055 | %FMGF0 | FLEX DISC FILE MNGR (APP DISC) | | 92064-13402 | 1709 |
| 92064-16056 | %DRF | F DISC DIRECT PROG (GEN DISC) | | 92064-13401 | 1650 |
| 92064-16056 | %DRF | F DISC DIRECT PROG (APP DISC) | | 92064-13402 | 1650 |
| 92064-16057 | %TBLFP | FLEXIBLE DISC DIRECT TABLES | | 92064-13401 | 1709 |
| 92064-16060 | %DRF1 | F DISC DIRECTORY SUB (GEN D) | | 92064-13401 | 1650 |
| 92064-16060 | %DRF1 | F DISC DIRECTORY SUB (APP D) | | 92064-13402 | 1650 |
| 92064-16075 | !MFGEN | ABSOLUTE FLEXIBLE DISC SYSTEM | | 92064-13401 | 1726* |
| 92064-16080 | %STRTM | RTE-M SYSTEM START-UP | 92064-13304 | 92064-13401 | 1709 |
| 92064-16081 | %MSYLB | RTE-M SYSTEM LIBRARY (GEN DISC | 92064-13306 | 92064-13401 | 1709 |
| 92064-16081 | %MSYLB | RTE-M SYSTEM LIBRARY (APP DISC | 92064-13306 | 92064-13402 | 1709 |
| 92064-18059 | %TBLCR | CARTRIDGE DIRECTORY TBLS SOURC | 92064-13306 | 92064-13402 | 1650 |
| 92064-18126 | %MHELP | EDITOR HELP FILE SOURCE | | 92064-13402 | 1650 |
| 92064-18141 | %MAUTO | AUTOR SOURCE | 92064-13306 | 92064-13402 | 1650 |
| 92064-18171 | %TBLFP | FLEXIBLE DISC DIRECTORY SOURCE | | 92064-13402 | 1709 |
| 92202-16001 | %DVR23 | RTE 7970 9T. MAG. TAPE DRIVER | 92062-13304 | 92064-13401 | A |
| 92900-16002 | %2DV47 | RTE 92900A DRIVER WITHOUT DMS | 92062-13302 | 92064-13401 | 1643 |
| 92900-16003 | %3DV47 | RTE 92900A DRIVER WITH DMS | 92062-13302 | 92064-13401 | 1643 |


# TRAINING SCHEDULE

The schedule for customer training courses on Data Systems Division products has been expanded to include courses offered at our European training centers. Listed below are courses offered in the U.S. and in Europe during the period May 1977 through August 1977.

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

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

## 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 accommodations reserved.

## ACCOMMODATIONS

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

## CANCELLATIONS

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

## TRAINING CENTER ADDRESSES

Cupertino

11000 Wolfe Road
Cupertino, California 95014
(408) 257-7000

Sunnyvale

974 East Arques
Sunnyvale, California

Rockville

4 Choke Cherry Road
Rockville, Maryland 20850
(301) 948-6370

Boise

P.O. Box 15
15 N. Phillippi Street
Boise, Idaho 83707
(208) 376-6000
TWX: 910-970-5784

Boblingen

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

Winnersh

King Street Lane
GB-Winnersh, Wokingham
Berks RG11 5 AR
Tel: Wokingham 784774
Cable: Hewpie London
Telex: 847178 9

Grenoble

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

Milan

Via Amerigo Vespucci, 2
1-20124 Milan
Tel: (2) 62 51
Cable: HEWPACKIT Milano
Telex: 32046

Madrid

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

Stockholm

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

TITLE

TRAINING COURSE RATES AND CENTER LOCATION

| Course Number | Length | Price | Cupertino | Sunnyvale | Rockville | Boise | Boblingen | Winnersh | Grenoble | Milan | Madrid | Stockholm | Amsterdam/ Brus. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01ETC | RTE II/III Driver Writing Course | | | | Jul 18 | | | | | | | | |
| | 3 days | 300 | | | | | | | | | | | |
| 22940A | 2100 Maint. | | | May 16<br>Jul 25 | | | | | | | | | |
| | 10 days | $1000 | | | | | | | | | | | |
| 22941A | 21MX Maint. | | | May 2<br>Jun 6<br>Jul 11 | | | | | May 9<br>Aug 8 | | | | |
| | 5 days | 500 | | | | | | | | | | | |
| 22942A | 7900 Maint. | | | May 9<br>Jun 20<br>Aug 8 | | | | | Jun 6<br>Sep 5 | | | | |
| | 5 days | 500 | | | | | | | | | | | |
| 22943A | 7970B Maint. | | | | | Jul 18 | | | Jun 20<br>Aug 29 | | | | |
| | 5 days | 600 | | | | | | | | | | | |
| 22944A | 7970E Maint. | | | | | Jul 11 | | | | | | | |
| | 5 days | 600 | | | | | | | | | | | |
| 22945A | 7905 Maint. | | | Jun 13<br>Jul 18 | | | | | Jun 13<br>Jul 25<br>Sep 12 | | | | |
| | 5 days | 500 | | | | | | | | | | | |
| 22950A | 2100 Ser. Assm. | | May 23<br>Jun 13<br>Jul 11<br>Aug 1<br>Aug 22 | | May 9<br>Jun 6<br>Jul 18<br>Aug 15 | | May 23<br>Jul 18<br>Aug 22 | Jun 13<br>Aug 8 | Jun 13<br>Oct 3 | May 30<br>Sep 5 | Jun 13 | May 23<br>Aug 22 | May 23<br>Sep 5 |
| | 5 days | 500 | | | | | | | | | | | |
| 22952A | DOS III B | | ** | | | | Aug 15 | | | | | | |
| | 5 days | 500 | | | | | | | | | | | |
| 22960A | 21MX Mic. Prog. | | May 9<br>Jul 18 | | | | | | Jul 18 | | | | |
| | 5 days | 500 | | | | | | | | | | | |
| 22965B | RTE-II/III | | May 9<br>May 16 | | May 16<br>May 23 | | Jun 20<br>Jun 27 | May 9<br>May 16 | May 9<br>May 23 | Jun 13<br>Jun 27 | Jun 20<br>Jun 27 | Jun 6<br>Jun 13 | Jun 6<br>Jun 13 |
| | 10 days | 1000 | May 16<br>May 23 | | Jun 13<br>Jun 20 | | Aug 1<br>Aug 8 | Jul 11<br>Jul 18 | Jun 20<br>Jul 4 | Sep 19<br>Oct 3 | | Aug 29<br>Sep 5 | Sep 19<br>Oct 3 |
| | (Course includes RTE-II/III operating system, batch spool monitor and file manager.) | | Jun 6<br>Jun 13 | | Jul 11<br>Jul 18 | | Aug 29<br>Sep 5 | Sep 26<br>Oct 3 | Sep 9<br>Sep 19 | | | | |
| | | | Jun 13<br>Jun 20 | | Jul 25<br>Aug 1 | | Sep 26<br>Oct 3 | | | | | | |
| | | | Jul 11<br>Jul 18 | | Aug 8<br>Aug 15 | | | | | | | | |
| | | | Jul 18<br>Jul 25 | | Aug 22<br>Aug 29 | | | | | | | | |
| | | | Jul 25<br>Aug 1 | | | | | | | | | | |
| | | | Aug 8<br>Aug 15 | | | | | | | | | | |
| | | | Aug 15<br>Aug 22 | | | | | | | | | | |
| | | | Aug 22<br>Aug 29 | | | | | | | | | | |
| 22968A | Measurement & control | | | | | | Sep 12 | | Sep 12 | | | | |
| | 2 days | 200 | | | | | | | | | | | |

TITLE

TRAINING COURSE RATES AND CENTER LOCATION

| Course Number | Length | Price | Cupertino | Sunnyvale | Rockville | Boise | Boblingen | Winnersh | Grenoble | Milan | Madrid | Stockholm | Amsterdam/ Brus. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22969A | Distb. Sys. | | May 16 | | Jun 6 | | Jul 4 | | Jun 6 | | | | May 9 |
| | 5 days | 500 | Jul 25 Sep 29 | | | | | | | | | | |
| 22977A | Image/DBMS 1000 | | Jun 6 | | Jul 11 | | May 9 | Jul 25 | | | Jul 4 | | |
| | 5 days | 500 | Jul 11 Aug 1 | | | | | | | | | | |
| 22978 | TCS | | * | | | | | | | | | | |
| | 2 days | 200 | | | | | | | | | | | |
| 22979A | Real/Time Multiterminal Basic | | ** | | | | Sep 14 | | Sep 14 | | | | |
| | 3 days | 300 | | | | | | | | | | | |
| 22980A | HPIB Multicomputer Bus Basic | | Jun 6 | | | | | | Jun 27 | | | | Aug 29 |
| | 3 days | 300 | Aug 8 | | | | | | | | | | |
| 22981A | HPIB Programming Under RTE | | Jun 9 | | | | | | Jun 30 | | | | Sep 1 |
| | 2 days | 200 | Aug 11 | | | | | | | | | | |
| 22983A | 21MX E-Micro-programming | | Jun 20 | | | | | | | | | | |
| | 5 days | 500 | Aug 29 | | | | | | | | | | |
| 22985A | RTE-M | | May 23 | | Aug 8 | | | | | | | | |
| | 5 days | 500 | Jun 20 Aug 8 | | | | | | | | | | |

*NOTE: Dates within brackets are starting dates for week 1 and week 2 of the RTE course. In some cases there is a break between the two weeks of the class. Course 22977A, IMAGE/DBMS 1000 replaces 22953A (2100 IMAGE); the new class adds additional material and extends the training from 3 to 5 days.

**On Sufficient Demand.

# COMMUNICATOR INDEX

## THE EDITOR

Below you'll find 2 Cross Reference Indexes for Past Communicators. The first lists all articles by Issue #, and the second by category. A key for the category abbreviations follow:

| | |
|---|---|
| BB | Bit Bucket |
| BU | BUlletins |
| CO | COmputation |
| HA | HArdware |
| IN | INstrumentation |
| OM | Operation Management |
| OS | Operating Systems |

Hopefully, you'll find these indexes valuable when requiring information discussed in past articles.

COMMUNICATOR ARTICLE INDEX
SORTED ON ISSUE #

| ISSUE # | CATE-GORY | ARTICLE TITLE | AUTHOR NAME | PAGE # |
|---|---|---|---|---|
| 1 | BB | HOW TO SPOOL IN TCS | PAUL MCGILLICUDDY | 4 |
| | | IMPLEMENT SELF-WRITTEN LOADR | JACK HOWARD | 2 |
| | | SHARING I/O SLOTS 10 & 11 | JACK HOWARD | 3 |
| | BU | CWF SUPPORT PLAN MODIFICATION | MARILYN BRANTHWAITE | 5 |
| | OS | DETERMINE PROGRAM LENGTH RTE-B | JOE DIESEL | 3 |
| | | RECON BCS FOR NEW INTERFACE | GEORGE TAYLOR | 5 |
| | | THE :ST.X DIRECTIVE | PAUL MCGILLICUDDY | 2 |
| | | WRITING DOS-3 DIRECTS TO FILES | R.K.STRAND | 2 |
| | | NUMBER OF ARTICLES THIS ISSUE = 8 | | |
| 2 | BB | CONCAT OF STRINGS IN HP BASIC | JEAN DANVER | 37 |
| | | HP FORTRAN OBJECT CODE GENERAT | LARRY W. SMITH | 41 |
| | | USING EXTENDED DCB BUFFERS | ERRYL JOHNSON | 45 |
| | BU | NEW PRODUCTS FOR RTE USERS | DAVE SANDERS | 47 |
| | OS | CENTRAL INTERRUPT REGISTER | GEORGE TAYLOR | 44 |
| | | DOS-3 LOGIC & PHYSICAL DRIVERS | PETER BAKER | 33 |
| | | HP 92002-16006 BATCH MON LIBR. | ERRYL JOHNSON | 48 |
| | | POWER FAIL AUTO RESTART RTE-B | JOE DIESEL | 47 |
| | | RTE-B MEMORY REQUIREMENTS | JOE DIESEL | 44 |
| | | RTE/BATCH SPOOL MONITOR | HAL SINDLER | 45 |
| | | NUMBER OF ARTICLES THIS ISSUE = 10 | | |
| 3 | BB | CONVERT SYSTEM DISC TO PERIPH | NORM WOLF | 110 |
| | BU | 3 PROGRAMS FROM CONTR. LIBRARY | PAUL MCGILLICUDDY | 110 |
| | | NUMBER OF ARTICLES THIS ISSUE = 2 | | |
| 4 | BB | DETERMINE OPTIMAL DCB SIZE | MIKE MANLEY | 175 |
| | | KNOW YOUR ASSEMBLER | LARRY SMITH | 182 |
| | | SOFTWARE SAM | SAM | 183 |

# COMMUNICATOR INDEX

# COMMUNICATOR INDEX

# COMMUNICATOR INDEX

COMMUNICATOR ARTICLE INDEX
SORTED ON SUBJECT-CATEGORY

| CATE-GORY | ISSUE # | ARTICLE TITLE | AUTHOR NAME | PAGE # |
|---|---|---|---|---|
| BB | 1 | HOW TO SPOOL IN TCS | PAUL MCGILLICUDDY | 4 |
| | | IMPLEMENT SELF-WRITTEN LOADR | JACK HOWARD | 2 |
| | | SHARING I/O SLOTS 10 & 11 | JACK HOWARD | 3 |
| | 2 | CONCAT OF STRINGS IN HP BASIC | JEAN DANVER | 37 |
| | | HP FORTRAN OBJECT CODE GENERAT | LARRY W. SMITH | 41 |
| | | USING EXTENDED DCB BUFFERS | ERRYL JOHNSON | 45 |
| | 3 | CONVERT SYSTEM DISC TO PERIPH | NORM WOLF | 110 |
| | 4 | DETERMINE OPTIMAL DCB SIZE | MIKE MANLEY | 175 |
| | | KNOW YOUR ASSEMBLER | LARRY SMITH | 182 |
| | | SOFTWARE SAM | SAM | 183 |
| | 5 | A PRIMER ON USING SPOOLING | JIM BRIDGES | 226 |
| | | INITIALIZE 21MX EIG INSTRUCTS | EARL STUTES | 229 |
| | | KNOW YOUR ASSEMBLER | MIKE MANLEY | 224 |
| | | REPLACE ON-LINE LOADR RTE2/3 | JIM BRIDGES | 225 |
| | | SOFTWARE SAM | SAM | 230 |
| | 6 | FEATURING DISTRIBUTED SYSTEMS | MIKE MANLEY | 256 |
| | | HP 7905 DISC BACKUP | MIKE MANLEY | 260 |
| | | SOFTWARE SAM | SAM | 260 |
| | 7 | ACODE PROB IN ALGOL,WORKAROUND | JIM BRIDGES | 315 |
| | 8 | NO ABORT RETURN;FORTRAN SUBRS. | JIM BRIDGES | 370 |
| | | PROGRAM SEGMENTING | JIM HOOPER | 365 |
| | | SOFTWARE SAM | SAM | 378 |
| | 9 | ASSIGN SSGA FROM FORTRAN | MARK SOLLE | 414 |
| | | PROGRAMMING W/ EMGR MACROS | JIM BRIDGES | 415 |
| | | SOFTWARE SAM | SAM | 421 |
| | 10 | DISTRIBUTED SYSTEMS TIMEOUTS | MIKE MANLEY | 476 |
| | | FORMAT OF DATA FILES IN MURB | JIM BRIDGES | 477 |
| | | FTN4 I/O USING ASSIGNS | DEL KITTENDORF | 480 |
| | | INDIRECT ADDRESSING | STEVE RUTEL | 476 |
| | | MULTIPLE CPUS & 7905 & RTE2/3 | JIM BRIDGES | 476 |
| | 11 | DEFINE 7905 SUBCHANNELS,RTE2/3 | JIM BRIDGES | 505 |
| | | INTRODUCING THE HP 1000 | GARY GUBITZ | 526 |
| | | SOFTWARE SAMANTHA | SAMANTHA | 509 |
| | 12 | 7905 DISC I/O OPTIMIZATION | MIKE MANLEY | 582 |
| | | SOFTWARE SAMANTHA | SAMANTHA | 586 |
| | | START PRESSING THOSE SOFT KEYS | GARY GUBITZ | 578 |
| | 13 | FAIL ERROR OPTION IN BASIC | JIM BRIDGES | 23 |
| | | KEEP PRESSING THOSE SOFT KEYS | GARY GUBITZ | 24 |
| | | SOFTWARE SAMANTHA | SAMANTHA | 25 |

NUMBER OF ARTICLES THIS SUBJECT-CATEGORY 39

# COMMUNICATOR INDEX

# 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: #_____ | | Your State & Local Sales Taxes* | | |
| If not, your order may have to be returned. | | 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.

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

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

**Printed in U.S.A. 7/77**

**Part No. 5951-6111**