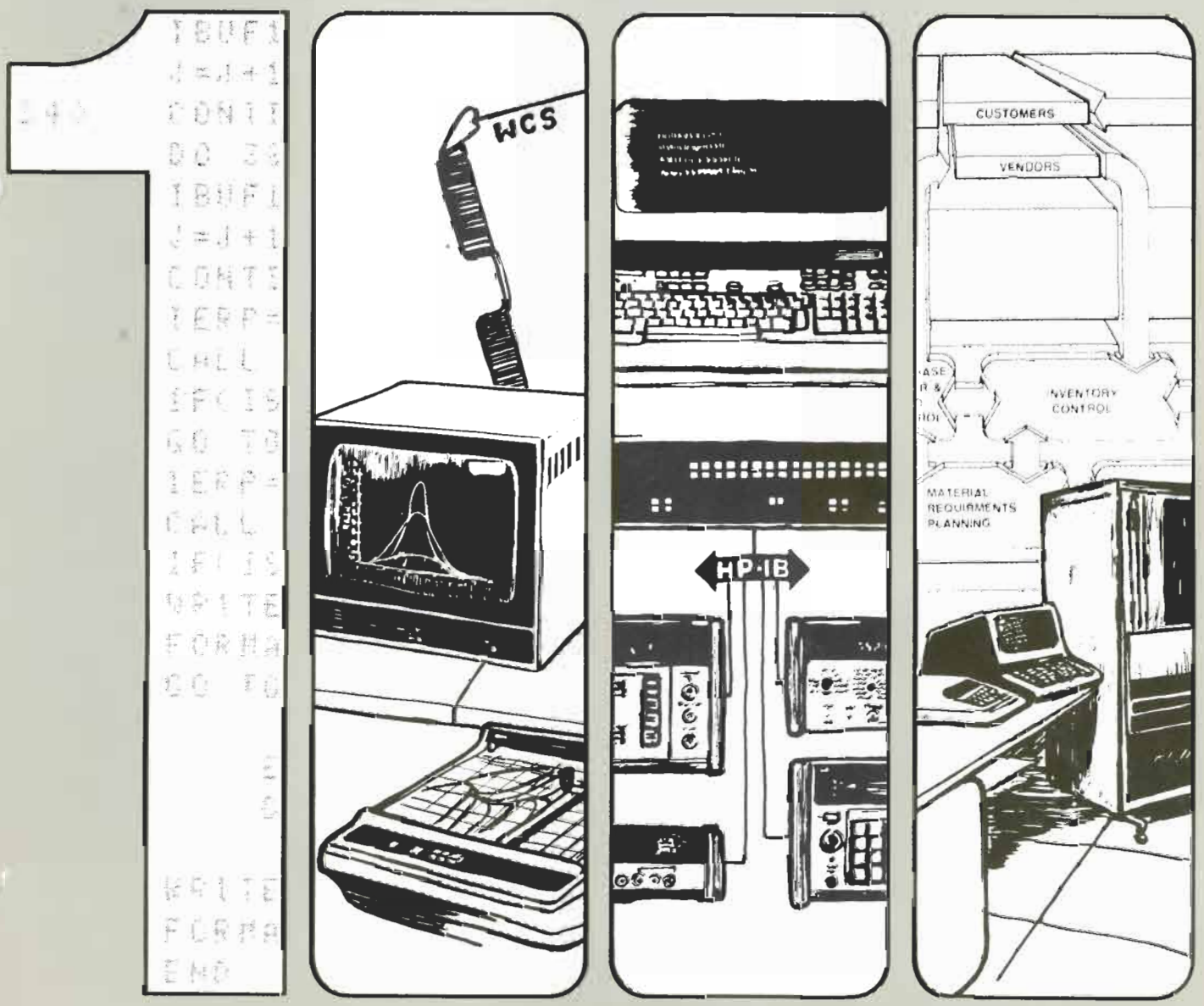
HEWLETT hp PACKARD

Computer Systems

COMMUNICATOR



issue no. 16

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

EDITOR'S NOTE

Remember LOCUS (Library of Contributed User Software). If you have any software you feel other customers might benefit from, become one of many who have added their software to LOCUS. Please see issue #15 for information and forms needed to become a contributor.

Please note that, starting this month, the COMMUNICATOR 1000 has a new feature. USER'S QUEUE is a new section we have added for you. If there is interesting information you would like to share with all of our other readers, USER'S QUEUE allows this to happen. From all of the contributed articles we receive from you, the two most interesting ones will be printed. For those articles chosen for issues #17 and #18 only, the contributing author can select either a LOCUS catalog ($15 value) or a program from the contributed library (cost not to exceed $20).

Thanks to those who contributed articles this month. It wasn't easy to select the two that we should print. They were all great.

Also in this issue . . . RTE hints, FORTRAN techniques, more time and date information, HP-IB and other informative items.

We at Hewlett-Packard are doing our best to keep you informed about the HP 1000.

Please address any correspondence to:

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

CONTENTS

In response to our request for technical articles, we have received some informative letters, and would like to pass the information along to you here.

The first article comes from J. A. Lorenz and A. J. Swierad, Jr. of Bell Laboratories, Holmdel, New Jersey:

"Using the program sequence described in the Communicator (pg. 592-593, Issue #12) presents a problem in that portions of the code may be executed only once. It is therefore necessary to implement flags to prevent the code from executing more than once. Described below is a technique which allows the code to be reused, eliminating the need for extra flags."

"The problem in the original program sequence was: assigning IB to IY(0) in line 6, destroyed null array IY's DEF location. Declaring two null arrays and equivalencing them, two DEF locations are generated. Now assigning IB to NOUSE(1) effectively changes the DEF location for array WRITE without altering its own DEF location. Array WHITE is then used to access the data in the FORMAT statement. By carefully adhering to these functional assignments for arrays NOUSE and WRITE, both variables may be used repeatedly."

"Two final notes, it is critical to follow the exact format specified in the example when declaring the INTEGER and EQUIVALENCE statements to generate the proper code. Also if the statement IY(0) = IB+3 is used, a one word overhead results in lieu of saying IY(0) = IB and using IY(3) in the Exec write call."

```
FTN4,M,L
      PROGRAM BSIGN
      INTEGER WRITE(0),NOUSE(0)
      EQUIVALENCE (WRITE(0),NOUSE(0))
      ASSIGN 100 TO IB
      NOUSE(1) = IB
100   FORMAT("THIS IS THE TEXT")
      CALL EXEC(2,6,WRITE(3),-16)
      END

      PROGRAM BSIGN
       00000  000000       NOP
       00001  016001X      JSB CLRIO
       00002  000003R      DEF *+1
       00003  026006R      JMP 00006
      INTEGER WRITE(0),NOUSE(0)
      EQUIVALENCE (WRITE(0),NOUSE(0))
      ASSIGN 100 TO IB
       00004  000004R      DEF *+0
       00005  000004R      DEF *-1
       00006  062010R      LDA *+2
       00007  002001       RSS
       00010  000020R      DEF 00020
       00011  072051R      STA IB
      NOUSE(1) = IB
       00012  062060R      LDA 00060
       00013  042052R      ADA 00052
       00014  042005R      ADA *-7
       00015  072053R      STA A.001
       00016  062051R      LDA IB
       00017  172053R      STA A.001,I
100   FORMAT("THIS IS THE TEXT")
       00020  026033R      JMP 00033
       00021  024042       ASC 1,("
       00022  052110       ASC 1,TH
       00023  044523       ASC 1,IS
       00024  020111       ASC 1, I
       00025  051440       ASC 1,S
       00026  052110       ASC 1,TH
       00027  042440       ASC 1,E
       00030  052105       ASC 1,TE
       00031  054124       ASC 1,XT
       00032  021051       ASC 1,")
      CALL EXEC(2,6,WRITE(3),-16)
       00033  062056R      LDA 00056
       00034  042052R      ADA 00052
       00035  042004R      ADA 00004
       00036  072053R      STA A.001
       00037  016002X      JSB EXEC
       00040  000045R      DEF *+5
       00041  000054R      DEF 00054
       00042  000055R      DEF 00055
       00043  100053R      DEF A.001,I
       00044  000057R       DEF 00057
      END
       00045  016002X      JSB EXEC
       00046  000050R      DEF *+2
       00047  000055R      DEF 00055
       00050  000000       OCT 000000
                           BSS 00001
       00052  177777       OCT 177777
                           BSS 00001
       00054  000002       OCT 000002
       00055  000006       OCT 000006
       00056  000003       OCT 000003
       00057  177760       OCT 177760
       00060  000001       OCT 000001


SYMBOL TABLE

NAME   ADDRESS USAGE              TYPE     LOCATION

@100   000020R STATEMENT NUMBER
CLRIO  000001X SUBPROGRAM         REAL     EXTERNAL
EXEC   000002X SUBPROGRAM         REAL     EXTERNAL
IB     000051R VARIABLE           INTEGER  LOCAL
NOUSE  000004R ARRAY(*)           INTEGER  LOCAL
WRITE  000004R ARRAY(*)           INTEGER  LOCAL
0010   END$
```

This second article is from John Blommers of the Defense Research Establishment Pacific, Victoria, British Columbia. Mr. Blommers has some interesting points to present on using the RTE Loader to scan the LG tracks for CALCOMP plotting routines not in the DOS/RTE relocatable library (Revision 1726). Here is what he has to say:

"...Revision 1726 of the DOS/RTE relocatable library [%RL1B1 & %RL1B2] does not contain the CALCOMP plotting routines PLOT, SYMB, NUMB, LINE, SCALE and AXIS.

Thus, the on-line LOADER will suspend itself for want of these routines (if they are required). Operator intervention is required to satisfactorily complete the load. The user must regain control of the FMGR and do a :MR, %PLTLB, a :EX, and a *GO, LOADR, 2, 0, 1 so that the LOADR scans the LG area like a library. The procedure will look like this:

```
:TR  #VPLT2
:RU, FTN4,&VPLT2, LIST %VPLT2, 58, LC
:LG, 1
:MR, %VPLT2
:RU, LOADR, 99, 6
       :
       :
LOADR SUSPENDS
• OF, FMGR
• RU, FMGR
:MR, %PLTLB
:EX
•GO, LOADR, 2,0,1
       :
       :
  LOADR COMPLETES
: :
```

The procedure I suggest is as follows:

```
:** SCHEDULE RTE FORTRAN IV
:RU,FTN4,&VPLT2,LIST,%VPLT2,20,LC
:LG,1
:MR,%VPLT2
:** SCHEDULE THE LOADR FROM THE SYSTEM
:SYRU,LOADR,99,6
:** RETURN  HERE IMMEDIATELY (DUMMY LINE)
:PU,XXXXX
:** LOADR NOW SWAPS OUT FMGR
:** LOADR EVENTUALLY SUSPENDS
:** FMGR REGAINS CONTROL HERE
:** MOVE THE PLOT LIBRARY
:MR,%PLTLB
:** RESCHEDULE LOADR TO SCAN LG AS A LIBRARY
:SYGO,LOADR,2,0,1
:** FMGR RETURNS HERE IMMEDIATELY
:** AND EXECUTES THIS DUMMY LINE
:PU,XXXXX
:** LOADR COMPLETES AND TERMINATES
:** FMGR REGAINS CONTROL HERE
:** AND RETURNS
::
```

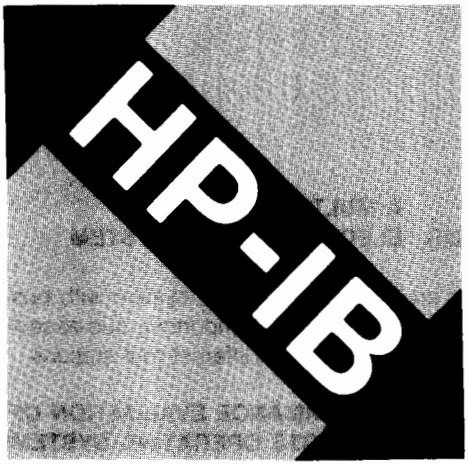
The :SYRU, LOADR directive does not cause the FMGR to schedule LOADR and thus waits until LOADR completes. The next directive, :PU, XXXXX, is executed as a dummy, allowing FMGR to be swapped by the LOADR. When LOADR suspends itself, FMGR is ready to: MR,%PLTLB. The whole transfer file executes hands-off."

Our thanks to J. A. Lorenz and A. J. Swierad, Jr. of Bell Labs, and to John Blommers of Defense Research Establishment Pacific for the fine information that we have been allowed to pass along to you.

If you have tips, techniques, applications or other technical information you feel would be beneficial to all our other readers, send it to:

EDITOR
COMPUTER SYSTEMS/COMMUNICATOR 1000
HP DATA SYSTEMS DIVISION
11000 WOLFE ROAD
CUPERTINO, CALIFORNIA 95014

Remember, for the two articles selected for publication in issue #17 and #18, the contributing author will have his or her choice of a LOCUS catalog or a program from the contributed library (cost not to exceed $20).



HP-IB TREKIE ARTICLE #6

HP-IB PERFORMANCE STUDY SUMMARY

Larry W. Smith/DSD

The HP-IB performance study as presented in the previous 5 issues of the COMMUNICATOR have indicated that the HP-IB handshake speed and other areas of consideration are dramatically affected when worst-case conditions prevail. In order to allow a designer to optimize the bus data rate under such conditions, you might find the following recommendations applicable to your environment:

1. Change the standard to permit the use of Schottky devices by allowing the driver output voltage to be 0.5 volt at 48 ma sink current.

2. Change the standard to allow a maximum device capacitance of 50 pf.

3. Change the standard to permit a data settling time of 350 nsec if tri-state drivers are used and the total cable length is limited to one meter per equivalent device load.

4. Change the wording of the standard to allow a bus attachment to present a dc load equivalent to multiple devices if it is known that the total number of attachments will be limited. This will permit a controller to simulate several best-case device loads. This best-case resistive load imparts the advantages of extra device attachements without introducing device capacitance, which affects open collector rise times and propagation delays.

5. Include some speed guidelines in the standard. These guidelines might take the form of figures shown in the article on 'Bus Topology & Handshake Analysis', perhaps including a curve showing the expected conditions rather than worst or typical cases.

6. Change the standard so the double DAV transition will not cause errors. Of the solutions listed in this report, the linear Topology/Schmitt trigger receivers might be the least painful.

7. Change the standard to require at least four-fifths the devices be powered on.

We hope that this series of articles about the HP-IB have been interesting and functional to you. Since the purpose of the articles was to arrive at system guidelines for optimizing HP-IB performance under worst case conditions, we also hope that the knowledgeable hardware designer can make use of the study for design purposes. All in all, we intend to make the HP-IB an understandable and easy to use interface system. If you have any questions regarding any of the articles, please write us. We will respond as quickly as possible to your request.

HP-IB PERFORMANCE BRIEF AVAILABLE

Neal Kuhn/DSD

A new performance brief, titled "Performance Evaluation of HP-IB using RTE Operating Systems" has been published. The brief covers the following topics:

* How HP-IB operates using RTE
* How to calculate measurement time
* How to calculate expected computer efficiency
* When to transfer data using DMA (direct memory access)
* When to transfer data using interrupt techniques

This performance brief presents a model which can be used to determine the time to send or receive data messages from various instruments and devices operating with the HP 1000 using RTE and HP-IB. The model can also be used to calculate the amount of spare time the computer will have during a measurement. This spare time, or unused computer potential, can be used to operate other HP-IB test stations or perform other program operations.

The following list shows the current HP-IB application notes available. They are all available from your HP salesman.

AN201-1 (5952-1578) ROUTINE QA MEASUREMENTS OF PRECISION RESISTORS

Describes an HP-IB based 21MX computer-controller instrumentation system capable of measuring, printing and plotting statistical distribution of precision resistor values.

AN201-2 (5952-9932) MEASURING DIFFERENTIAL NON-LINEARITY OF VOLTAGE CONTROLLED OSCILLATOR

Describes an HP-IB based 21MX computer-controlled instrumentation system for measuring and plotting differential non-linearity of the modulation sensitivity of a voltage controlled oscillator.

AN201-3 (5952-1686) A MULTIPLE STATION ELECTRONIC TEST SYSTEM

Describes an HP 1000 computer system with bus-connected instruments for component, sub-assembly, and final product tests at three different test stations.

AN201-4 (5953-0864) PERFORMANCE EVALUATION OF HP-IB USING RTE OPERATING SYSTEMS

This brief contains performance data to help determine whether the Hewlett-Packard Interface Bus is suitable for various interface applications. A model is developed to help the HP-IB user determine the total time to send or receive a data message and the amount of HP 1000 or 21MX computer utilization. Performance examples with various devices, such as the HP 3455 digital voltmeter and the HP 2240 measurement and control processor are included.

AN INTRODUCTION TO DATA BASE MANAGEMENT TERMINOLOGY

Gary McCarney/Rockville

INTRODUCTION

A data base is a collection of information that has been stored in such a way that easy access to the information is made possible. The designer of a data base must decide well in advance how this data should be stored so that retrieval is rapid and easy. In order to meet these goals, certain fundamental rules must be observed. These rules are written using terminology common to data base management systems. This terminology is confusing to many people, particularly scientific programmers. The purpose of this article is to explain this terminology beginning with the smallest entity and gradually building to a complete description of a data base. By using an inventory control example, the structure of the IMAGE 1000 Data Base Management System (DBMS) will be examined. No previous data base background is assumed of the reader.

DATA ITEMS

The smallest entity in a DBMS is known as a *data item*. For example, the vendor name of some part would be a data item, the part name would be another data item, etc. As our inventory control system is designed, some amount of space (storage) must be allocated for storing the contents of each data item. Assuming our inventory information is stored on punched cards, we might assign the first ten columns for vendor name, the next 20 columns for part number, and so on. Then the first data item uses ten columns and the second data item uses 20 columns.

When it becomes necessary to find the card which contains a part number 1997643325 from vendor WIDGETS, we need a way to specify which columns on each card are to be searched. Consequently, each data item is assigned an *attribute* which contains the necessary information about card columns and length. The IMAGE 1000 DBMS can have up to 255 unique attributes using up to six characters each. For example, let the attribute VENDOR have a length of ten columns starting at column one, and attribute PARTNO be 20 columns long starting at column eleven. Now we can search for a VENDOR of WIDGETS and a PARTNO of 1997643325.

The actual values that can be stored into data items can be integer, real or ASCII. Integer values use five columns and are denoted in the DBMS as an "I1." Real values are defined as ten column fields and denoted as R2. Finally, ASCII strings are indicated by 'Ux' where 'x' is any even number of characters up to and including 126.

DATA ENTRY

The next level in the DBMS system is the *data entry* which is a collection of data items. All the data items on a punched card might be defined as a data entry.

VENDOR U10	PARTNO U20	QTY I1
WIDGETS	1997643325	00005

Figure 1. Data Entry Example

Figure 1 shows a data entry consisting of three data items having attributes VENDOR (10 ASCII characters maximum), PARTNO (20 ASCII characters maximum) and QTY (one word integer using five columns). For those readers familiar with the RTE file management package (FMP) terminology, a data entry is similar to an FMP record. The maximum size permitted for a data entry is 512 bytes.

DATA SETS

A collection of similar data entries is known as a *data set*. Figure 2 contains four data entries which together make up a particular data set that will be referred to as the INVENTORY data set. The maximum size for each data set is 32767 data entries. Data sets are similar to FMP files.

VENDOR U10	PARTNO U20	QTY I1
WIDGETS	1997643325	00005
AUTOS	4599200043	00001
AXES, INC	0000056477	00358
BWD ENT	0506722381	10000
WIDGETS	9006773265	00047
AXES, INC	0788004537	00500

Figure 2. Data Set Example

FINDING PARTICULAR VALUES

One of the reasons for building this inventory is to be able to determine how many items are in stock. Perhaps we get a request to find all items that exist in stock from a particular vendor. It would be necessary to search all the contents under the VENDOR attribute to find all entries for this vendor. This requires a sequential search through the complete inventory. If some type of relationship could be maintained to prevent the need for a sequential search each time we seek inventory information from a vendor, then the search time could be significantly reduced.

Let's decide to include within each data entry a pointer from Vendor A to the next data entry that references Vendor A. Figure 3 illustrates such a pointer setup.

VENDOR U10	PARTNO U20	QTY I1
WIDGETS	1997643325	00005
AUTOS	4599200043	00001
AXES, INC	0000056477	00358
BWD ENT	0506722381	10000
WIDGETS	9006773265	00047
AXES, INC	0788004537	00500

Figure 3. Linking Similar Data Item Contents

Since VENDOR is the data item within each data entry that should contain such a pointer, this particular data item is known as the *key item*. Therefore, all key items are linked together by similar contents. Now when a search is required for a particular vendor, it is not necessary to search the entire collection of data entries, rather simply follow the pointers (known as forward pointers or links). How do we know when we have found the last entry in the data set? The pointer must contain some special character to indicate the end, such as zero.

At some point, it becomes necessary to add another part number to our inventory for Vendor A. We search all linked entries for Vendor A until we find the link that contains a zero. We then replace the zero with a pointer to the entry that we are adding. The new entry will automatically get a forward pointer of zero. There will be times when we change to another vendor for a particular part and wish to remove a particular data entry for the old vendor. Once we find the appropriate entry, how do we know which pointer needs to be modified? The link we find on the data entry to be deleted only points to the next occurrence of this vendor. Therefore, we need to modify the previous data entry's link.

One way to solve this problem is to include not only a link to the next entry (a forward link) but also a backward link to the previous entry.

Now when an entry is removed, we simply modify the next record with the backward point from the entry we seek to delete, and in turn modify the forward pointer of the previous entry to bypass the entry we are deleting. These forward and backward pointers will require storage within the data entry. This storage is known as the *media record*.

The media record is considered part of the data entry and contains both forward and backward pointers for each key item within the data entry. There can be at most five key items per data entry — that is, there can be forward and backward links for, at most, five different data items within the data entry. The contents of the link is simply a relative record pointer as shown in Figure 4.

Record Number	Backward Link	Forward Link	VENDOR	PARTNO
1	0	5	WIDGETS	199 . . .
			. .	
5	1	0	WIDGETS	900 . . .
			. .	
13			empty	

ORIGINAL

Backward Link	Forward Link	VENDOR	PARTNO
0	5	WIDGETS	199 . . . .
		. .	
1	13	WIDGETS	900 . . . .
		. .	
13	0	WIDGETS	678 . . .

AFTER ADDITION

Backward Link	Forward Link	VENDOR	PARTNO
0	13	WIDGETS	199 . . . .
		. .	
		WIDGETS	900 . . . .
		. .	
13	0	WIDGETS	678 . . .

AFTER DELETION

Figure 4. Media Records

Now it is only necessary to perform a sequential search to find the first occurrence of a particular key item. Then using the links, access may be made directly to the next desired data entry. To further reduce the access time, we need some quick way to find the first occurrence of the desired data item.

FINDING THE FIRST ENTRY

In order to find the first data entry in a list, it may be desirable to maintain a separate data set which contains a single data entry for each key item. For example, a set that contains only vendor names. The media record for this data entry would contain a pointer to the first occurrence of this vendor in another data set. Since this separate data set is being developed to reduce the time required to find the first occurrence of this vendor, how does a separate data set reduce the access time? It would appear that a serial search in either data set will require about the same amount of time.

The data items in this separate data set will not be stored in sequential fashion. Instead, the data items will be stored into relative records whose addresses are calculated by performing a mathematical routine upon the contents of the data item itself, a method known as *hashing*. When our sample vendor WIDGETS is processed through the hashing routine, perhaps the relative record address that is calculated is five. Now every time some type of access is required involving the key item WIDGETS, the DBMS will process the data item through the same hashing routine to determine the relative record in the separate data set. This entry will contain a pointer to the first occurrence of this vendor within the second data set. See Figure 5.

Assume this additional data set is called the NAMES data set. It will be used for each access to the INVENTORY data set. One question that often comes up is — why have two data sets? Why not hash the entries into the INVENTORY data set directly? Considering our example, we have two part numbers from vendor WIDGETS. If we stored in the INVENTORY data set by using the hashing routine on the vendor name, both part numbers would try to be stored at the same location. There is no easy way to store one and still reference the other part number.

Since entries in the NAMES data set are stored into relative locations determined by hashing, there may be times when two or more vendor names are similar enough to hash to the same relative location. The first time a data entry location in the NAMES data set is filled, it is known as a *primary* entry. The next time hashing points a new entry to this record, it is known as a *synonym* or *secondary* "hit". Since we cannot store more than one piece of information about a vendor within each NAMES data set entry, it is necessary to "flag" the fact that a synonym has occurred. This flag is stored

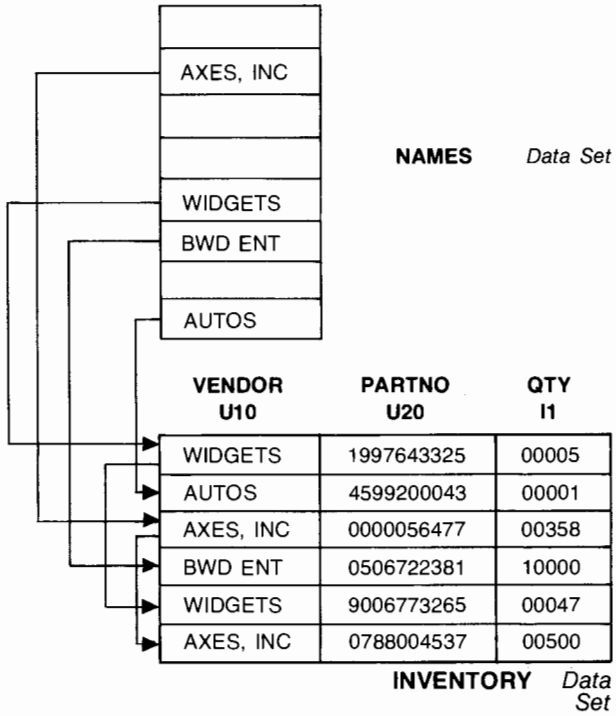


Figure 5. Data Set Relationships

within the primary data entry. Synonyms are then stored in the first available data entry after the primary.

For example, assume we are adding a new vendor SERIAL. Further assume that the address calculated after SERIAL has been hashed is two. Location two is already filled. (See Figure 6.a.) The next available record is number three and SERIAL is stored there (see Figure 6.b.). A pointer is set up from record two to record three indicating a synonym.

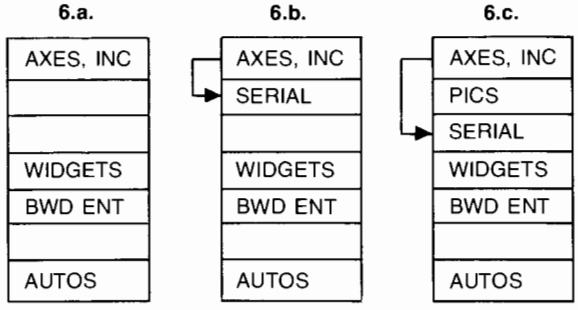


Figure 6. Primary and Secondary Entries

As a new primary entry (vendor PICS) hashes to record that is currently being occupied by a secondary entry (say record three), the following takes place: First, the secondary

entry is moved to the first empty data entry and the pointer from its corresponding primary is changed. Then the new primary entry replaces the record that had been used by the secondary entry (see Figure 6.c).

Each data entry in the NAMES data set will also require a media record. This media record contains pointers to the first and last occurrence of the key item in the INVENTORY data set as well as synonym pointers, if any.

Each access takes us to a particular data entry in the NAMES data set where a check is made to see if we have the right vendor. If not, the media record is checked for synonyms. When a match is found, we have a pointer to the first data entry in the INVENTORY data set. For additions to the INVENTORY data set, the last occurrence pointer provides the relative data entry location. The zero forward pointer is replaced with the new data entry location and the last occurrence pointer is updated.

TYPES OF DATA SETS

The data sets containing the hashed entries are known as *master data sets*. There are two types of master data sets: automatic and manual masters. The automatic master contains only the media record and a single data item which is the key item. Manual masters contain the media record, the key item, and can contain other data items as well. More details about the differences between the automatic and the manual master will be covered in the next section.

The data sets that contain the vendor name, part number, quantity, etc. are called *detail data sets* since they contain all the details about some particular inventory component.

MASTER DATA SETS

The master data sets contain media records that point to the corresponding data entry (key item) in detail data sets. Each master entry can contain pointers to as many as five different detail data sets. These pointers are referred to as *data paths* — links from masters to details.

Automatic master data sets contain data entries consisting of a media record and a key item only. Any additions to a detail data set that contains key item attributes pointed to by an automatic master will force a new entry into the master for each new key item value. Deletions from the master will be done automatically when the last detail data entry is removed. Automatic masters are error prone since a new entry could result from transposed digits, misspelled vendor name, etc.

Manual master data sets consist of a single key item but can also contain non-key data items as well. Perhaps for each vendor in a manual master data set, we have the vendor as the key item and the vendor's address as the non-key items. Additions of detail data entries to a data set that has key items pointed to by a manual master requires the user to first make additions to the manual master for new vendor names before it is possible to add the detail entry. Therefore, all additions and deletions from a manual master data set involve two operations when a new key item is concerned.

For example, assume there is a remote terminal in the receiving department which is used to communicate with our inventory data base. When a shipment of components arrives, the receiving clerk types the part number, vendor name and quantity received. If the clerk incorrectly types the part number, what happens to the data base?

If we have defined the part number as a key item in a manual master, the DBMS will search the master for this part number. When the number is not found, an error will be reported to the clerk. Now the clerk can correct the entry. However, if the part number was a key item in an automatic master, the DBMS would create a new master entry for the incorrectly typed part number.

DETAIL DATA SETS

Detail data sets contain a media record and one or more data items for each data entry.In our example, INVENTORY is a detail data set. All key items within the data entry will require pointers within the media record. The media record can contain information about five key items per data entry. The media record contains forward and backward links to data entries for a similar key item entry. These links are known as *data chains*.

As we create multiple detail data sets and master data sets, these data sets are collectively known as a *data base*. With the IMAGE 1000 DBMS there is a possibility of having a total of 50 different data sets which include masters and details. Figure 7 contains a summary of the capacities permitted in an IMAGE 1000 data base.

50	data sets per data base
32,767	data entries per data set
512	bytes per data entry
255	different data items per data base
126	bytes per data item
6	characters per data item name
5	keys per detail data set
5	detail data sets per master data set

Figure 7. Data Base Capacity (Maximum Value)

WHAT IS A DATA BASE?

As we have seen, a data base is merely a collection of information that has been structured into a preferred form to permit rapid access to the contents. As the information is stored into the data base, no ordering is necessary since the links provide the ordering. The manager of the data base must decide well in advance what type of structure is desired for this data base and design it accordingly. All chain maintenance is performed automatically by the IMAGE 1000 Data Base Management System. See the HP IMAGE/1000 Data Base Management System Reference Manual, (Part No. 92063-90001) for the necessary steps for this design and subsequent usage.

This article is intended to assist in the understanding of the terminology that is used throughout that manual and in most other data base discussions. Hopefully, the reader can now understand and use these new terms.

A SOLUTION TO THE RTE MULTI-TERMINAL BLUES

Larry Smith/DSD

If you are an RTE user and have ever wondered why there is only one logical source (LS) and one load-and-go (LG) area, making activities such as program development from a multi-terminal standpoint frequently inconvenient and more often than not frustrating, then this article is a must for you to read. The answer to this bewildering question requires a historical review of the HP Real-Time Executive Operating System from its beginning to the present date of this article, RTE-III. The evolution of any operating system is like observing style and price changes of an automobile from one year to another — you take what you have and improve (enhance) it until you're convinced it meets the demands of the market. Then you hope to make some money. RTE has been a proven system for many years on a large and quite diversified customer base by maintaining a constant policy of upgrade.

You might further ask yourself, "Is there a short term cost-effective solution to this problem without investing a lot of time and resources into applications programming?" It will be the main purpose of this article to present a typical solution that has been thoroughly tested in hopes that it might solve your problem and/or give you some additional insight on how to cope with the situation. All in all, whatever your exposure has been to RTE, you might be able to pick up some concepts that will help you in other areas as well.

RTE- PAST AND PRESENT

The original RTE in 1968 ran on a 2116A computer with a 7901 or 7900 disc and was called "RTE-E". The operating system had all the basic properties of a Real-Time system as the current version with the exception of some recent enhancements such as Class I/O, Resource Numbers, Subsystem Global, LU lock, queue scheduling, and some program interface capabilities. The system was designed for single-terminal program preparation where input and output to several devices such as terminals was a normal built-in capability of the system.

Furthermore, since File Manager was in development stages at that time, the requirement for multi-terminal program preparation was not in large demand by the majority of our customer bases. If other terminals were configured into the system, they were used for data input and display.

Current with the development and announcement of the original FMP, the predecessor of the current Multi-Terminal Monitor (MTM) was a program called "AUXTY". The program ran by taking advantage of one of the four possible ways to schedule a program under RTE.

1. Operator — RU,FMGR
2. Time of Day — IT,WHZAT,2,5; ON,WHZAT,NOW
3. Program — CALL EXEC (9,NAME,IP1,. . .)
4. External Interrupt (strike key on terminal and get prompt)

EXTERNAL INTERRUPT

This required a special entry into the interrupt table during system generation (SC,PRG,AUXTY) and an interface routine to the system message processor ($MESS) so that one additional terminal would have the same capability as the system console for entering operator commands.

As the popularity for AUXTY increased and the announcement of FMP, which had some system related commands such as "RP" and "SP", AUXTY was given a facelift and redesigned into two programs, PRMPT and RPN. These routines used later system enhancements such as class I/O to give the user the capability of executing the same program at different terminals. The only remaining problem was, and still is, the LS and LG areas which are nothing more than scratch work areas for the standard utilities. This problem was left for future resolvement since it would mean a somewhat extensive overhaul of the operating system, FMP, and all utilities including the system generator. Thus, variable procedure files were implemented in a later enhancement of FMP as a tool for the user to develop a multi-terminal system tailored to a specific application. This is the one we intend to explore.

THE OBJECTIVE — A SESSION ORIENTED RTE

Our ultimate objective is to make RTE function like a multi-terminal system where each terminal runs a standard procedure file for activities such as program preparation, editing, etc. Since there are as many ways to solve this problem as there are to optimize a FORTRAN program, the solution presented in this article utilizes only one capability of FMP:

Variable Procedure Files

and is not intended to be as comprehensive as the HP 3000 system, nor is it intended to be the best optimal solution. So, sit back, grab your favorite computer, and evaluate.

To make a program operate in a multi-terminal environment, you must first consider how it is written:

Case	Type of Program
1	Main not using LS or LG.
3	Main using LS and/or LG.
2	Main with segment(s) not using LS or LG.
4	Main and segment(s) using LS and LG.

The structure and considerations for your procedure file will vary slightly according to the type of program.

Case 1: Main Program and segments not using LS or LG.

This is probably the simplest case since there is no contention for LS and LG and no segments called by the main. Using the system Real-Time editor, EDITR, as an example, we can construct a generalized variable procedure file that will give the terminal user the following:

Terminal LU=1

```
:RU,EDIT
SOURCE FILE?
/
 .
 .
 .
/ER
END OF EDIT
 .
```

Terminal LU=7

```
:RU,EDIT
SOURCE FILE?
/
 .
 .
 .
/ER
END OF EDIT
 .
 .
```

Everybody runs the same name and the FMP procedure file does the rest.

To implement this basically involves two steps:

Step A: Set-Up a copy of the editor within FMP under a different name.

Step B: Design a procedure file named "EDIT"

Step A can be solved by entering the following command sequence once:

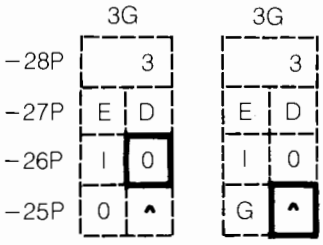
```
:SP,EDITR:SC:-2::-1
:RN,EDITR:SC:-2, EDI00
```
(This will be the "Master Copy")

We will later see that it is advantageous to load the editor on-line, save in FMP, and not make available in system area for a solution to programs using LS and LG. With the editor renamed to EDI00, step B can now be done by creating an efficient FMP procedure that will construct a copy of EDI00's ID segment for each terminal as follows:

```
:RU,EDIT[,rec-length]
```
rec-length=maximum record length for editor
(recall,default=150 characters)

Remembering that FMP preserves the terminal LUN in 0G (−40 P thru −37P the procedure file looks like this:

```
1. :SV,4,9,IH          Save existing security code in 9G
                       and set to no command echo.
2. :CN,0G,21B          Disenable terminal interrupts so user
3. :LL,0G              doesn't prematurely abort this file.
4. :CA,3,EDI00         Set in master copy name in 3G.
5. :LO,0               Set log LU to bit bucket to suppress
                       all possible FMP errors (step 10)
6. :CA,-25:P,0,-39P,/,10,t,-39P,*,400B,+,-25P
```



Form the ASCII equivalent of the binary terminal LU in 0G into 3G.

```
7. :CA,-26:P,-39P,/,10,+,-26P
8. :RP,,3G             Get rid of EDI0G in case of a
                       previous procedure file abort.
9. :CA,6:P             Set in no command error (6P=0).
10. :RN,EDI00:SC:-2,3G Attempt a re-name.
11. :IF,6P,NE,0,-3     If a rename not in progress from
                       another terminal, then proceed to
                       restore program.
12. :RP,3G:SC:-2       Create ID segment.
    :IF,6P,NE,0,8      Check for no available ID
                       seg-system overload.
13. :RN,3G:SC:-2,EDI00 Rename back to original for other
                       pending terminals.
14. :RU,3G:SC:-2,0G,1G Run EDI0G.
15. :RP,,3G            Release ID segment.
16. :RT,3G             Release work area tracks.
17. :CN,0G,20B         Re-enable terminal for interrupts.
18. :SV,9G             Restore severity code.
19. :SE                Null all globals for next entry.
20. :TR
21. :AN,0G,System Overload - Task canceled
22. :IF,,EQ,,-6
```

Although the above procedure file has the advantage of being general purpose for any program name of the format:

xxx00
where xxx=any three ASCII characters

it inherently contains the characteristics of any procedure file; that of being slow and requiring a few disc accesses such as in lines 9 - 11. In most session oriented applications, the former outweighs the latter since real-time response is usually not important.

The previous example centered around the idea of allocating and releasing program ID segments on a first-come, first-serve basis, regardless of priority and resources. The idea was to minimize the number of required blank ID segments needed for on-line activities in a further effort to optimize system table space. If, on the other hand, your system memory requirements are slim, then a modified approach can be taken in the procedure file that will be faster and more efficient.

The first step is to allocate all ID segments in the WELCOM file at system bootstrap time:

```
:RN,EDITR:SC:-2,EDI01
:RP,EDI01:SC:-2
:RN,EDI01:SC:-2,EDI07
:RP,EDI07:SC:-2
   .
   .
   .
```

With this done only once, then its a simple matter of forming the ASCII program name in 3G and running the ID segment in memory.

To modify this for Case 2 would simply involve inserting as many "RP,NAME" commands for all program segments (with appropriate checks) into the command file after line 14 or make them available in the WELCOM file. The modified transfer file would look like this.

```
:SV,4,9,IH
:CN,0G,21B
:LL,0G
:CA,3,EDI00
:LO,0
:CA,-25:P,0,-,-39P,/,10,+,-39P,*,400B,+,-25P
:CA,-26:P,-39P,/,10,+,-26P
:CA,6:P
:RU,3G:SC:-2,0G,1G
:IF,6P,EQ,0      Ensure ID segment still there.
:AN,0G,EDITOR NOT AVAILABLE FOR THIS TERMINAL
:RT,3G
:SV,9G
:SE
```

Since there is less to worry about in this example, the response time will significantly improve at the terminal and the disc will not be as busy. The only disadvantage is the vulnerability of the ID segment to the "OF" command (OF,ED10G,8).

Case 2: Main Program and Segment(s)

The only functional difference between this and case 1, is the fact that all segments of a main must be defined in short ID segments when the main is run. Keeping further in mind that segments can be shared by more than one program because the segments are swapped into the main, it is suggested that all segments be allocated at system bootstrap time. Since there is no convenient means to allocate and release short ID segments in a procedure file and due to the fact that they only occupy 9 words apiece of system table space, the justification to leave them always allocated becomes practical. Thus, simply add a few commands to the WELCOM file or a file of your choice. As an example, let's use Real-Time BASIC.

```
:RP,BASC1
:RP,BASC2
   .
   .   } Only 72 words of system table space
   .
:RP,BASC8
```

Now replace lines 4, 10, and 13 with BAS00 and you have a procedure file for Real-Time BASIC.

Case 3 & 4: Main program with or without segments using LS and/or LG

Since this type of program uses the scratch work areas, the procedure file actually becomes simpler:

```
:SV,4,9,IH
:CN,0G,21B
:LO,0G
:LL,0G
:CA,6:P
:RP,!FTN4:SC:-2
:IF,6P,NE,0,15   Check to see if another terminal
                 running program
:RP,F4.0
:RP,F4.1   } Allocate ID segments
:RP,F4.2
:RP,F4.3
```

```
:RU,!FTN4,1G,2G,3G,4G,5G
:RT,!FTN4
:RP,,F4.0
:RP,,F4.1
:RP,,F4.2
:RP,,F4.3
:RP,,!FTN4
:CN,0G,20B
:SV,9G
:SE
:TR
```

Release ID segments

```
:AN,0G,FORTRAN COMPILER BUSY-TASK CANCELLED
:IF,,EQ,,-6
```

With this concept, you can further expand this into a more general purpose procedure file by defining it's usage as follows:

```
:RU,FTN4,source-input,list-output,
 reloc-output,security[,LOAD]
```

SOFTWARE SAMANTHA
Messrs. Software Samantha
Care of Communicator
1000 (9600) Group
HP Data Systems Division

Dear Samantha:

I saw your note in the Communicator (p. 25, issue No. 14), regarding use of function IGET in lieu of array IGET. I had previously noted that techniques for array base storage and element address calculation (herein after referred to as "techniques") are different in the new FORTRAN compiler. Unfortunately function IGET cannot be used to solve all problems users may have generated by taking advantage of the techniques as implemented in the old compiler (for example, Gary Gubitz's short program (p. 592 issue No. 12).

The old techniques allowed one to take all sorts of shortcuts and unhappily I have done just that in over 100 programs. I was quite aware that I was not using standard FORTRAN but reasoned that in the "unlikely event" that HP should switch to a FORTRAN implementation which used different techniques for array base storage and element address calculation, I would simply install both compilers in our RTE system and convert as routine program maintenance was required. Alas, HP was able to close that door by using the same segment names in the new compiler that they use in the old one. It is not feasible for me to revise all of the programs and I don't have sufficient disc storage to save them as relocatable files.

Can you help me? Specifically, if I could get the source code for the old compiler, I could rename the segments and install both compilers. Any other suggestions would be appreciated as I would sincerely like to install the new compiler, but will require the old one also for quite some time.

Yours Very Truly,
Jeff Wynne
Code 3021, Bldg. 759
USN Ordnance Station
Indian Head, MD 20640

Dear Jeff,

There exist a couple easy ways of installing both compilers on a system for compilation of old programs which take advantage of the methods of array storage and address calculation used by the old compiler. The user may save both compilers in type six files using the SP/RP File Manager commands and then rename as necessary, or supply his or her own copy of the program SEG.F which returns a pointer to the segment name given the segment number.

Let us consider first the alternative of saving and restoring the main and segments with the File Manager SP and RP commands. The first step is to load the new compiler into the system on-line or at generation. Using SP, <program> save the main and segments in type six files on LU 2. If loaded at generation a RU,LOADR,,,4 will be required to delete the main and segments from the permanent program area. Next do a temporary load of the old compiler on-line. At this point a naming problem exists.

One option available is to save the old compiler on LU 3 and create the necessary transfer files to restore and release the ID segment and tracks of the main and segments.

Another option at this point is to rename the existing program files and save the old compiler on LU 2. User may then create transfer files which restore and release ID segment and tracks and also do the necessary renaming of files before restoring. For example, consider the set of transfer files below:

```
 /FTN4O  (RESTORE OLD FTN4)
:RN,FTN4O,FTN4
:RN,F4.00,F4.0
:RN,FR.10,F4.1
:RN,F4.20,F4.2
:RN,F4.30,F4.3
:RP,FTN4
:RP,F4.0
:RP,F4.1
:RP,F4.2
:RP,F4.3
:RN,FTN4,FTN4O
:RN,F4.0,F4.00
:RN,F4.1,F4.10
:RN,F4.2,F4.20
:RN,F4.3,F4.30
:TR
```

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

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

A second method of installing both compilers involves modifying the program SEG.F, the function of which is to return a pointer to the name of the segment, given the segment number. For example, given the value N, SEG.F returns a pointer to the string "F4.N". A modified SEG.F which returns a pointer to the name "F5.N" for value N is shown below:

```
ASMB,R
       NAM SEG.F
       ENT SEG.F
* PURPOSE:  GIVEN SEGMENT NUMBER
*     RETURN POINTER TO SEGMENT
*     NAME.
*
*               JSB SEG.F
*               DEF SEG#
*          -> RETURN  B POINTS AT NAME
*
SEG.F  NOP
       LDA SEG.F,I    GET DEF TO SEG#
       ISZ SEG.F      GET RETURN ADDRESS
       LDA A,I        GET SEGMENT NUMBER
       ADA ".0"       CONVERT TO ASCII AND ADD "."
       STA NAM+1      SET IN NAM STRING
       LDB PTR        RETURN POINTER
       JMP SEG.F,I    TO STRING IN B
*      *
PTR    DEF NAM
NAM    ASC 3,F5.X
".0"   ASC 1,.0
A      EQU 0
       END
```

After the modifications have been made and the new compiler loaded and saved the user can then rename the RP files to match:

```
:RN,FTN4,FTN5
:RN,F4.0,F5.0
:RN,F4.1,F5.1
:RN,F4.2,F5.2
:RN,F4.3,F5.3
:RN,F4.4,F5.4
```

Next load and save the old compiler and finally, for user convenience, create transfer files which restore and replace the ID segments and program tracks.

```
 /FTN4
:RP,FTN4
:RP,F4.0
:RP,F4.1
:RP,F4.2
:RP,F4.3
:TR
```

```
 /FTN5
:RP,FTN5
:RP,F5.0
:RP,F5.1
:RP,F5.2
:RP,F5.3
:RP,F5.4
:TR
```

```
 /FTN4
:RP,,FTN4
:RP,,F4.0
:RP,,F4.1
:RP,,F4.2
:RP,,F4.3
:TR
```

```
 /FTN5
:RP,,FTN5
:RP,,F5.0
:RP,,F5.1
:RP,,F5.2
:RP,,F5.3
:RP,,F5.4
:TR
```

Thus, user may install both compilers by a number of different methods. User may store the second compiler on the auxiliary LU 3 or rename program files before restoring the program. Or, provide your own version of SEG.F as shown above, and rename program files to correspond.

If you have any questions or comments about your 1000 (9600) system please address them to:

SOFTWARE SAMANTHA
c/o Communicator Editor
Hewlett-Packard
Data Systems Division
11000 Wolfe Road
Cupertino
CA 95014

SOFTWARE REVISION CODES

Dick Walker/DSD

Understanding the meaning and use of the Software Revision Codes appearing on released software and its supporting documentation is highly desirable for efficient operation of your installation. Such awareness has the potential of preventing confusion and many hours of needless program debugging.

Every HP 1000 software module has an associated Software Revision Code that identifies the release date for either new software or a subsequent enhancement to the software module. This number (e.g., 1740) appears on a label fixed on the outside of the distribution media; for instance, fixed on grandfather discs for RTE-II and III systems, or on the paper cover of flexible discs for RTE-M flexible disc systems. For mini-cartridge versions of RTE-M, the Software Revision Code may be found by reading the File Directory, which is the first file on any mini-cartridge containing HP-supplied software.

The Software Revision Code also appears in the NAM record of each module. NAM records can be found in the loader or generation listings for the software. The Software Revision Codes in these listings appear in the form:

REV. xxxx (e.g., 1740)

and are immediately followed by a six-digit internal HP release code that has no particular relevance for end users.

It is critical to good system housekeeping that only the updated software is accessible to users. It is suggested that older versions of software modules retained for special applications be kept away from the work areas when not in use.

Whenever new or updated software is distributed to users on the Software Subscription Service, new or updated documentation is also supplied in one of the three forms: a new manual, a change package (added and/or changed pages for an existing manual), or a complete revision. In every case, the documentation also contains a Software Revision Code that reflects the software being described. The number can be found in a manual or manual change in the following places:

a. Manual title page, where it will appear as a message similar to the following:

   "(This manual reflects information that is compatible with 92064A Software Revision Code 1740)."

b. Under "Publication History" on the Publication Notice page located on the back of the title page. The last change listed will reflect the Software Revision Code of the latest software. The "Publication History" will also list all manual changes for the current edition of the manual.

c. On the top right-hand side of the "Manual Change Notice" cover page for a manual change package.

Good preventative housekeeping requires that updated documentation replace older versions as soon as it is received, and that the Software Revision Code of a manual always match the code on the software being used!!

PROGRAMMATICALLY UPPING A DEVICE IN RTE

Larry W. Smith/DSD

You say, "How could RTE ruin my personal life?" Well, that's exactly what could happen if you were to get called out of bed by the graveyard operator, drive 30 miles to work, just for the thrill of entering the 'UP,eqt' command on the system console. The coupler controller can now resume operation. Sound bizarre? Well, those of you with rings under your eyes might appreciate the information presented in this article. For sake of brevity, this story is authenic and the individual who lives it is not alone.

The above incident was relayed to me by an HP employee at our Stanford Park Division in vivid detail and I was asked for advice on the matter. This individual wanted to know how to make sure that the coupler controller (DVR66) would always remain up regardless of the state of the device. The purpose of this article is to present a solution to this problem and refresh your memory on how the RTE system and driver handle I/O requests.

A LITTLE I/O REVIEW

Let us briefly review situations in which a device could be declared down by RTE before we discuss a solution to the problem. If you'll recall, a device is initially declared down by the driver rejecting an I/O request due to a bad initial status return from the interface. This further assumes that the device has the capability of returning status bits so the driver can determine whether it is powered down, not ready, or otherwise. If the driver determines that the device is unavailable and needs operator attention, then RTE downs either the LU (RTE-III) or the EQT (RTE-II) and then prints one of the following messages indicating the reason for reject on the system console (LU=1):

	DEVICE CONDITION...	MESSAGE...
1.	Not Ready	I/O NR L 6 E 4 S 0
2.	Parity Error	I/O PE L 8 E 7 S 2
3.	End-of-Tape	I/O ET L 5 E 2 S 1
4.	Time-Out	I/O TO L23 E10 S24

If the device does not have appropriate status line(s) (such as the paper tape reader), then the driver must rely upon its status posted in bits 0-6 of EQT word 5 on previous interrupts or time-outs to evaluate possible errors and line consistency. In addition, the driver can also set bit 14 of EQT word 5 to indicate that the device is down so that RTE can put other programs requesting I/O in the general wait list. If the driver chooses to do this and then issue a normal continuation return to RTE, no message is printed on the system console, but the device will still be downed by RTE next clock interrupt. All in all, whenever RTE determines that the device is down by examining bit 14 of EQT word 5, it puts the calling program into the general wait list (state=3), making it swappable, and then envokes the dispatcher to put the next highest priority program into execution. The net result is that operator intervention or a programatic call to MESSS or $$CMD is required to resume program execution and put the device back into operation. If, on the other hand, the driver chooses to send a reject code back to RTE instead to indicate device unavailability, RTE prints one of the above messages and downs the LU or EQT. In RTE-III, the LU is downed but the EQT remains up unless the driver sets bit 14 of EQT word 4 or the operator enters the 'DN,eqt' command. In either RTE-II or RTE-III, the EQT must be declared up.

THE PROBLEM

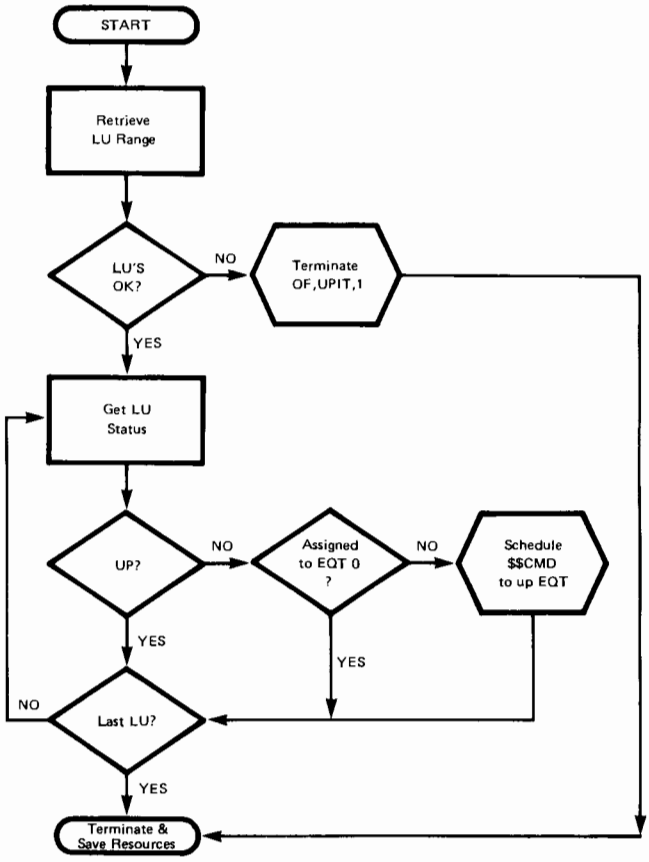
We need to construct a method of ensuring that a device will always remain available (UP) such that operator intervention can be eliminated. The solution presented in this article is primarily for RTE-III and must be modified slightly relative to the EQT for RTE-II.

A SOLUTION

Let's assume that a program is periodically requesting I/O on a device in which its driver is capable of detecting a line failure and informing RTE that the device is to be declared down. Furthermore, lets also assume the device can be put into this state at any given time by the operator for such things as off-line usage or simply a power-down or disconnect. Since the driver must be envoked in order to determine whether it is available, a status request from the program would not be able to determine device availability. Recall, a status request (code=13) does not call the driver but RTE simply returns words 4 and 5 of the EQT and the LU status (RTE-III ONLY). Thus, the only SAFE solution would be to write an independent program which is scheduled periodically to UP the LU if it went down. Let's call this program a 'line monitor' with the program name

UPIT

and give it the sole responsibility of guaranteeing that the LU is always UP. Its flow would be as follows:



Now, by specifying this routine as a memory-resident program or reserving it to run in a specific partition (RTE-III ONLY) to avoid scheduling and swapping overhead, you have yourself a method of ensuring the stability of one or more lines.

The program 'UPIT' would look like this:

```
FTN4,L
C
      PROGRAM UPIT(1,10)
C
C  THIS PROGRAM IS SCHEDULED PERIODICALLY, RETRIEVES A
C  RANGE OF SYSTEM LU'S, AND ENSURES THAT THEY ARE ALL UP.
C
C  PROGRAM SCHEDULE:
C
C      IT,UPIT,res,mult[,hour[,min[,sec[,ms]]]]
C
C      ON,UPIT,NOW[,start-LU[,ending-LU]]
C
C         start-LU ---> start lu to check (default=none)
C         ending-LU ---> ending lu        (default=none)
```

```
C
      DIMENSION LUN(5),ICMND(3)
C
      EQUIVALENCE(LUN1,LUN(1)),(LUN2,LUN(2)),(LUN3,LUN(3))
      EQUIVALENCE(LUNAD,LUN(4)),(NLUN,LUN(5)),(ICMND1,
     &ICMND(1))
C
      DATA ICMND/2H$$,2HCM,2HD /
C
C... RETRIEVE LU RANGE & VALIDATE RANGE ...
C
    1 CALL RMPAR(LUN1)
      NLUN=IGET(1653B)
      IF(LU1.GT.0.OR.LUN1.LE.NLUN.OR.
     &-LUN2.GT.0.OR.LUN2.LE.NLUN) GO TO 10
C
C... BAD NEWS (ONE OR MORE LU'S BAD) TERMINATE & REMOVE FROM
C    TIME LIST ...
C
      ITERM=3
      GO TO 999
C
C... GET STATUS OF EACH LU & IF DOWN, UP IT ...
C
   10 LUNAD=IGET(1652B)-1
      DO 200 LUNN=LUN1,LUN2
      LUNADJ=LUNAD+LUNN
   90 IF(IGET(LUNADJ+NLUN))150,200
C
C... IT IS DOWN QUEUE SCHEDULE $$CMD AND PASS 'UP,eqt'
C    COMMAND ...
C
  150 IEQT=IAND(IGET(LUNADJ),77B)
      IF(IEQT.EQ.0) GO TO 200
      CALL EXEC(23,ICMND1,2HUP,IEQT,-1,-1,1)
  200 CONTINUE
C
C... TERMINATE PROGRAM EXECUTION ...
C
      ITERM=1
  999 CALL EXEC(6,0,ITERM,LUN1,LUN2)
      GO TO 1
C
99999 END
```

For RTE-II, you would simply pass on EQT range upon schedule and send 'UP,eqt' commands to $$CMD in which case 'LUSTAT' (computed by 'LUNAD+NLUN' in line 41 of 'UPIT') would not be available and bit 14 of EQT word 5 would have to be examined instead.

CONCLUSION

The above solution is implemented entirely in FORTRAN IV which relies upon an external integer function 'IGET' that is in the RTE library to retrieve the contents of memory, namely system base page locations 1652 (DRT address) and 1653 (#DRTS) to avoid making a call to MESSS. Although the program is systems-oriented and optimized to minimize system overhead, it illustrates still another example of the flexibility and control the user can exhibit on-line in RTE.

HOW TO USE CLASS I/O AND RESOURCE NUMBERS IN A SORT APPLICATION

Jim Bridges/DSD

Class I/O and Resource Numbers are two powerful concepts which often are not used on the System 1000 due to lack of understanding. The sample programs "GETEM" and "SORTM" included in this article illustrate and comment upon these concepts.

The basic task which this example addresses is a familiar one — that of sorting records. The fastest and simplest sort is performed on an array in memory. This places a limit on the amount of data that can be sorted to that which will fit into memory. It also forces the programmer to consider how to miminize the memory used by the sorting procedure itself so that as much as possible can be used for the data. In addition, there must be some way of transmitting the sorted output to a program which can use it: presumably, this program is not attached to the sort procedure and the data array because that would limit the data array even further.

Class I/O and Resource Numbers are powerful tools in addressing this problem. The technique is to separate the sort procedure and the data into a program with minimum overhead code. The "mailbox" scheme described in the RTE reference manual is the method used to pass data between the sort program and any other program which can obtain or use the data. The mailbox scheme employs a class number (a unique tag for the data) and the area of memory called "SAM" (system available memory) as a buffer accessible to co-operating programs. The resource number concept is used to synchronize programs so no program monopolizes SAM and thus makes the system "sluggish". SAM is used in a number of ways that are transparent to the programmer and usually there are limits to the amount of memory a program can use up. (Memory is, in most cases, "used up" only temporarily and returned when not needed.) Class I/O gives a programmer a greater flexibility in the amount of SAM used (he can use it all, if he choses) but places greater emphasis on using it "intelligently".

The sample programs include several comments to help follow the code. "GETEM" is a "skeleton" program which would be "fleshed out" according to the user application. It allocates a resource number locally (meaning only this program is entitled to release the number) and locks it globally (meaning other programs can lock and unlock the number). The feature which permits synchronization of data transfer is the action taken when a lock request is made with a lock already on. In this case the second lock request causes a suspension of the requesting program. The global nature of the lock permits another program to unlock the number and restart the program. In this example, GETEM performs both lock requests, the second just prior to writing a buffer to SAM via class I/O. If this technique (or equivalent) were not used, GETEM could fill up all of SAM before SORTM were ready to pick up any data at all. In the interim, the entire system would be "bogged down". Perhaps, due to priorities of competing programs, SORTM would never execute and then a deadlock could occur.

A similar synchronizing sequence is built into SORTM, which has the capability to transmit the sorted data through class I/O (however, a program to receive is not included). Note that SORTM has four separate tasks, which are selected individually by the scheduling program (the "father"). Three of these tasks result in a termination saving resources. This means that, unless task 4 is specified, that the data can be restored on several keys or returned from SORTM several times after the array has been filled because it (the array) remains intack after going dormant.

The actual sort procedure is called as a subroutine (SORT) from SORTM. There are numerous ways to sort in memory but one of the simplest and fastest is the SHELL sort technique. A particular version of this routine used by the author (coded in assembly language) is included because many people learning on the System 1000 do not have the time to research sorting techniques. It is hoped that it will prove valuable.

It is important to note that arrays are stored sequentially by column in memory. When passing a first word address (e.g., BUF(1,I)) you are passing a column — not a row of data. This is also important when using the SORT procedure shown.

```
FTN4,L
      PROGRAM GETEM (3,45),GET DATA AND PASS TO SORTM
      INTEGER P(5),RN,ERR,CLASS,REC(12),LU,LIST,LEN,OFSET,TASK,
     &CNWD
C
      EQUIVALENCE (LU,P(1)),
     &            (LIST,P(2))
C
      CALL RMPAR (P)
C
C ALLOCATE RN LOCALLY AND LOCK GLOBALLY
C THEN GET CLASS NUMBER. GET DATA AND PASS TO SORTM
C
      CALL RNRQ (12B,RN,ERR)
C
      CLASS = 0
      LEN = 12
      OFSET = 0
      TASK = 1
      CNWD = 0
      I1 = 0
C
C GET CLASS NUMBER
C
      CALL EXEC (20,CNWD,REC,12,I1,I2,CLASS)
C
C RELEASE BUFFER SENT OUT AND KEEP CLASS NUMBER
C
      CLASS = IOR (CLASS,20000B)
      CALL EXEC (21,CLASS,REC,12,I1,I2,I3)
      CALL EXEC (24,6HSORTM ,LEN,OFSET,CLASS,RN,TASK)
      DO 500 I = 1,N
C
C  INSERT CODE TO GET DATA FROM YOUR SOURCE
C  LOCK RN TO SUSPEND YOURSELF (DOUBLE LOCK) UNTIL SON SORTM
C  RELEASES LOCK. THEN SEND DATA TO SORTM.
C
      CALL RNRQ (2,RN,ERR)
500   CALL EXEC (20,CNWD,REC,12,I1,I2,CLASS)
C
C   TAG I1 NON-ZERO TO INDICATE DONE WITH DATA TO SORTM
C
      I1 = 99
      CALL EXEC (20,CNWD,REC,12,I1,I2,CLASS)
C
C  RELEASE RESOURCE AND CLASS NUMBERS AND TERMINATE SORTM
C
      CALL RNRQ (40B,RN,ERR)
      CLASS = IAND (CLASS,157777B)
      CALL EXEC (21,CLASS,REC,12,I1,I2,I3)
      CALL EXEC (24,6HSORTM ,LEN,OFSET,CLASS,RN,4)
      END
      END$
```

```
FTN4,L
      PROGRAM SORTM (3,45),PERFORM IN-MEMORY SORT
      INTEGER P(5),LEN ,OFSET,CLASS,RN,TASK,CNWD,ERR,REC(12,500)
C
      EQUIVALENCE (LEN ,P(1)),
     &            (OFSET,P(2)),
     &            (CLASS,P(3)),
     &            (RN,P(4)),
     &            (TASK,P(5))
C
      CALL RMPAR (P)
50    GO TO (100,200,300,400) TASK
C
C  TASK = FILL UP ARRAY RECEIVED FROM FATHER
C         UNLOCK RN (FATHER HAS LOCKED TO SUSPEND) TO SYNCHRONIZE
C         AND GET DATA THROUGH MAILBOX (CLASS) I/O
C
100   I = 0
      CNWD = 0
110   I = I + 1
      IF (I.GT.500) GO TO 700
C
      CALL RNRQ (4,RN,ERR)
      CALL EXEC (21,CLASS,REC(1,I),12,I1,I2,I3)
      IF (I1.EQ.0) GO TO 110
      N = I - 1
      GO TO 900
C
C  TASK = SORT AND EXIT
C
200   CALL SORT (REC(1,1),N,12,OFSET,LEN )
      GO TO 900
C
C  TASK = TRANSMIT ARRAY BACK TO FATHER. LOCK RN UNTIL FATHER
C         UNLOCKS TO SYNCHRONIZE
C
300   DO 350 I = 1, N
      CALL RNRQ (2,RN,ERR)
350   CALL EXEC (20,CNWD,REC(1,I),I1,I2,CLASS)
      GO TO 900
C
C  TASK = TERMINATE AND CLEAR RESOURCES
C
400   CALL EXEC (6)
C
C  "UNLESS SPECIFICALLY INSTRUCTED OTHERWISE, TERMINATE
C   SAVING RESOURCES AFTER EACH TASK
C
900   CALL EXEC (6,0,1)
      CALL RMPAR (P)
      GO TO 50
C
C  ERROR REPORT ON LU #1: TOO MANY RECORDS PASSED
C
700   CALL EXEC (2,1,8HSORTM ER,4)
      CALL EXEC (2,1,8HBF OVFL ,4)
      END
      END$
```

```
ASMB,R,L
      HED SHELL SORT WITH HIBBARD MODIFICATION TO IMPROVE SPEED
*
*  CALL SORT (ARRAY(1,1),COLS,ROWS,OFSET,LEN)
*
*     ARRAY (1,1) = FIRST WORD OF ARRAY TO SORT
*     COLS        = NUMBER OF COLUMNS
*     ROWS        = NO OF ROWS (FIELD LENGTH)
*     OFSET       = NO ROWS OFFSET INTO RECORD TO START SORT
*                   FIELD (ZERO EQUAL NO OFFSET)
*     LEN         = NO ROWS TO INCLUDE IN SORT FIELD (SIZE OF
*                   ITEM FOR COMPARISON TEST)
*
      NAM SORT,7
      ENT SORT
      EXT .ENTR,.MVW
BUFR  NOP           ARRAY TO SORT
SIZE  NOP           SIZE OF ARRAY
FL    NOP           FIELD LENGTH
OF    NOP           OFFSET INTO FIELD
KS    NOP           SIZE OF ITEM FOR COMPARISON
SORT  NOP           ENTRY POINT
      JSB .ENTR
      DEF BUFR      GET PARAMETERS
      LDA FL,I
      STA FLEN      SAVE FIELD LENGTH
      LDA KS,I
      STA CNT       SET COUNT FOR COMPARE TEST
      LDA OF,I
      STA OFSET     SET OFFSET
      LDA SIZE,I
      STA N         SET SIZE IN N
      CMA,INA
      STA MN        MN = - M
      CLB,INB
      STB A
```

```
      RAL           FIND LARGEST POWER
      RBL           OF 2 THAT IS
      ADA MN        LESS THAN N
      SSA
      JMP *-5
*
      ADB M1        SUBTRACT ONE
      STB M
SETM  LDB M
      BRS
      STB M
      SZB,RSS
      JMP SORT,I
*
COMPK LDA M
      CMA,INA
      ADA N
      STA K         K = N - M
      LDA ONE
      STA J         J = 1
LOOPI STA I         I = J
      ADA M1
      SSA           DONE WITH I?
      JMP TESTJ     YES. TEST IF DONE WITH J
*
      MPY FLEN      MPY BY FIELD LENGTH
      ADA BUFR
      STA ADDR1     SAVE ADDRESS LOW RECORD
      LDA I
      ADA M
      ADA M1
      MPY FLEN
      ADA BUFR
      STA ADDR2
      JSB SWAP      SWAP IF NEEDED AND RETURN
      LDA M         COMPLEX RECORDS.
      CMA,INA
      ADA I
      JMP LOOPI     I = I - M
*
TESTJ LDA J
      CPA K
      JMP SETM
*
      ISZ J         J = J + 1
      LDA J
      JMP LOOPI
*
SWAP  NOP           ENTRY POINT FOR SWAP SUBROUTINE
      CLA
      STA CTR       COUNTS WORDS FOR COMPARE TEST
SWAP1 LDA ADDR1
      ADA OFSET
      ADA CTR
      LDB ADDR2
      ADB OFSET
      ADB CTR
      LDA A,I
      LDB B,I
      CMA,INA
      ADA B
      SZA,RSS       ARE THEY EQUAL?
      JMP NEXT1     YES, TRY NEXT WORD
      SSA           NO. IS HIGH REC > LOW REC?
      JMP SWAP2     NO. DO SWAP
      JMP SWAP,I    YES. NO SWAP NEEDED.
NEXT1 ISZ CTR
      LDA CTR
      CPA CNT
      JMP SWAP,I    DONE WHEN HAVE COMPARED # WORDS IN CNT
      JMP SWAP1
SWAP2 LDA ADDR1
      LDB BFR
      JSB .MVW
      DEF FLEN
      NOP
      LDA ADDR2
      LDB ADDR1
      JSB .MVW
      DEF FLEN
      NOP
      LDA BFR
      LDB ADDR2
      JSB .MVW
      DEF FLEN
      NOP
      JMP SWAP,I
*
FLEN  NOP           FIELD LENGTH
CNT   NOP           COUNT OF WORDS TO COMPARE
CTR   NOP           TEMP COUNTER TO MATCH TO CNT
OFSET NOP           OFSET IN FIELD TO BEGIN COMPARE
N     BSS 1
I     BSS 1
J     BSS 1
K     BSS 1
ADDR1 BSS 1
ADDR2 BSS 1
M     BSS 1
MN    BSS 1
M1    DEC -1
M2    DEC -2
ONE   DEC 1
BFR   DEF TBUF
TBUF  BSS 40        THIS IS LIMIT OF SWAP BUFFER
A     EQU 0
B     EQU 1
      END
$
```

EXPANDED CAPABILITIES FOR NEW DRIVER

Melanie Fox/DSD

A versatile new driver featuring full-duplex asynchronous modem support for Bell 103 or equivalent (Vadic 3400) type modems is now available.

DVA05 (part number 92001-16035) is a modified version of driver DVR05 (to be used with Software Revision Code 1740 or greater) that provides all the capabilities of DVR05, plus modem support. The Multi-Terminal Monitor can be used for program development on user terminals with hardwired DVA05 or DVA05 over modems. However, only hardwired DVA05 can be used with the system console — DVA05 over modems is intended for use only with user terminals.

Both hardwired DVA05 and DVA05 over modems can communicate with the RTE system in both ASCII or binary codes.

DVA05 is approximately 250 words larger than the largest version of DVR05. DVR05 ranges from approximately 900 to 1350 words (depending on the terminal model) while the size of DVA05 is about 1600 words for all terminal models (2635, 2640, 2644, 2645, and 2648).

Calling formats for drivers DVR05 and hardwired DVA05 are identical. There are, however, three additional requests that allow modem support with DVA05.

Information and procedures for writing FORTRAN or Assembly Language applications programs that call either of the drivers (DVR05 or DVA05) can be found in the RTE Drivers DVR05/DVA05 for HP 263x/264x Terminals manual (part number 92001-90015).

DVA05 will be supplied to all Software Subscription Service customers, and can also be ordered as an independent part by those customers that are not on the SSS list (contact your local HP Sales Office for more information).

FILLING STRINGS IN FORTRAN ARRAYS

Jim Bridges/DSD

Often it is desired to initialize an array with ASCII characters. Typically, the array will contain an error message or a prompt to be issued to a terminal. Perhaps the array may be initialized to key words to be used as search patterns over an arbitrary field of text.

Most programmers will use DATA statements with 2H format, e.g.,

```
DATA IBUF/2HTH,2HIS,2HI,2HS,2HA,2HST,2HRI,2HNG/
```

which is "THIS IS A STRING". This method of breaking down strings into two character fields is tedious and makes the code difficult to read. Two new features added to the FORTRAN IV compiler (which was recently revised and issued under a different part number, 90206-16092) make the job easier:

- Ability to use up to 8 characters in the H format
- Named COMMON

If you use the extended H format, you must be careful to match the size of the H field to the data type. The following is a common error:

```
INTEGER IBUF (8)
DATA IBUF/8HTHIS IS ,8HA STRING/
```

Since the data type is integer, only the 2H field can be used. The following EQUIVALENCE enables the programmer to use the full 8H format:

```
INTEGER IBUF (8)
COMPLEX BUF (2)
EQUIVALENCE (BUF,IBUF)
DATA BUF/8HTHIS IS ,8HA STRING/
```

Since a complex variable has four words, 8H may be used to fill the array. The COMPLEX type declaration was used only so that the 8H format could be used and the individual elements of the array will never actually contain complex numeric quantities.

The limit on the number of words initialized at a time may be removed entirely by using named COMMON and writing the BLOCK DATA "subroutine" in assembly language rather than FORTRAN. (This scheme requires very little knowledge of assembly language). Consider the following FORTRAN program and the symbol table printed by the compiler:

```
PAGE 0001 FTN. 10:48 AM TUE., 10  AUG., 1976

 0001  FTN4,L,T
 0002        PROGRAM TRYIT
 0003        COMMON/L1/MSG1(20)/L2/MSG2(20)
 0004        CALL EXEC (2,15,MSG1,20)
 0005        CALL EXEC (2,15,MSG2,20)
 0006        END

 FTN4 COMPILER: HP92060-16092 REV. 1726

      ** NO WARNINGS ** NO ERRORS **
    PROGRAM = 00022      COMMON = 00000

PAGE 0002 TRYIT 10:48 AM TUE., 10 AUG., 1976

 SYMBOL TABLE

 NAME  ADDRESS USAGE          TYPE     LOCATION

 CLRIO 000001X SUBPROGRAM     REAL     EXTERNAL
 EXEC  000002X SUBPROGRAM     REAL     EXTERNAL
 L1    000003X COMMON LABEL   INTEGER  EXTERNAL
 L2    000004X COMMON LABEL   INTEGER  EXTERNAL
 MSG1  000000+ ARRAY(*)       INTEGER  L COMMON L1
 MSG1  000000+ ARRAY(*)       INTEGER  L COMMON L2

PAGE 0003 FTN. 10:48 AM  TUE., 10 AUG., 1976

 0007        END$
```

The following assembly language subroutine will fill in the arrays MSG1 and MSG2:

```
ASMB,R,L
      NAM LINES,7 DATA FOR FORTRAN STRINGS
      ENT L1,L2
L1    ASC 14,THIS IS MY MESSAGE FOR MSG1
      ASC 6,DATA ARRAY!!
L2    ASC 14,THIS IS MY MESSAGE FOR MSG2
      ASC 6,DATA ARRAY!!
      END
      END$
```

Named COMMON is treated quite differently than blank COMMON in FORTRAN. The "name" creates an external symbol (in this case, L1 and L2) but does not allocate storage for the associated variables. Hence, the assembly language "subroutine" (not really called or entered as subroutine) merely provides data to be linked into this storage area by the LOADR (or generator).

The ASC pseudo operator in the assembler code is limited to a field of 20 words. However, since multiple ASC's can be coded sequentially, there is no practical limit to message length.

JULIA AND JULIS, SYSTEM TIME/ DATE ROUTINE

Alan Tibbetts/DSD

It is often convenient to have listings or other hardcopy dated, so that confusion will be kept to a minimum when software is updated at short intervals. For example, the RTE FORTRAN compiler places the system time in the header of each page.

The RTE system keeps track of the time, and will even share it with you. The time is kept in memory at system entry point $TIME. The first two words are the number of 10's of milliseconds until midnight (in 2's complement), and the next word is the day of the year and the year in a binary format. This is very handy if you wish to compute elapsed time, just read $TIME before and after the event and take the difference. (Remember, it is in 10's of milliseconds.)

The other way the system time can be accessed is by an EXEC call. The RTE time request, CALL EXEC (11,ITIME [,IYEAR]), returns the array;

ITIME(1) = 10's of milliseconds
ITIME(2) = Seconds
ITIME(3) = Minutes
ITIME(4) = Hours
ITIME(5) = serial day of year (e.g., 253)
IYEAR = Year (e.g., 1975) (optional)

Although this is good because the values are separated for you, it is in a format which must be converted to something else before it is useful to human beings. The following FORTRAN callable routine will return the time in a ready to print (A2) format. Note that the routine can be assembled to return the time to the second or to the minute, depending upon your needs. Although the routine as written is useable only on MX or XE, the Contributed Library has an emulator package to help those of you with the older machines. (22682-18965 on paper tape, or 22682-13365 on mini-cartridge)

```
ASMB,R,N,L,C
      HED JULIAN TIME ROUTINES   1SEP77
      IFZ
      NAM JULIA,7 TIME ROUTINE A.T. 1SEP77 <<21MX ONLY!>>
      ENT JULIA
      XIF
      IFN
      NAM JULIS,7 TIME ROUTINE A.T. 1SEP77 <<21MX ONLY!>>
      ENT JULIS
      XIF
      EXT EXEC,.ENTR
*
*   THIS PROGRAM CONVERTS THE RTE ORDINAL DAY TIME TO "MILITARY"
*   TIME IN ONE OF TWO FORMATS.  THE TIME AND DATE IS THEN PRINTABLE IN
*   A2 FORMAT.  THE VALUES OF THE DAY AND MONTH ARE ALSO RETURNED IN THE
*   A AND B REGISTERS.
*
*               ASSEMBLY OPTION Z              ASSEMBLY OPTION N
*               FOR TIME TO MINUTES            FOR TIME TO SECONDS
*
* AMSB CALL-->   JSB JULIA                       JSB JULIS
*                DEF *+2                         DEF *+2
*                DEF TBUF                        DEF TBUF
*              <RETURN POINT>                  <RETURN POINT>
*
* FTN4 EXAMPLE>  PROGRAM TEST                    PROGRAM TEST
*                DIMENSION ITBUF(6)              DIMENSION ITBUF(8)
*                CALL JULIA(ITBUF)               CALL JULIS(ITBUF)
*                CALL ABREG(IDAY,IMONTH)         CALL ABREG(IDAY,IMONTH)
*                WRITE (LU,1000)ITBUF            WRITE (LU,1000)ITBUF
*          1000  FORMAT("TIME="6A2".")     1000  FORMAT("TIME="8A2".")
*                END                             END
*
*   OUTPUT-->    TIME=2305 06AUG76.     OR       TIME=23:05:15 06AUG76.
*
*  ON RETURN:  A=DAY OF THE MONTH          B=MONTH# (1 TO 12)
*              X=L.S.D. OF YEAR            Y=NEXT BYTE ADDRESS OF OUTPUT BUFFER
*
*
* WRITTEN BY:  ALAN TIBBETTS
*              HEWLETT-PACKARD DATA SYSTEMS DIV.
*              CUPERTINO, CA.
*
MSEC  NOP
SECS  NOP
TEMP  EQU SECS
MINS  NOP
HRS   NOP
DAYS  NOP
YEAR  NOP
*
OBUF  NOP
JULIA EQU *
JULIS EQU *
RETRN NOP
      JSB .ENTR      GET RETURN ADDRESS
      DEF OBUF
*
      JSB EXEC       GO SEE WHAT TIME IT IS        ***
      DEF *+4                                       *
      DEF D11                                       *
      DEF MSEC                                      *
      DEF YEAR                                      ***
*
* NOW TRANSFER STUFF TO USER
*
      LDB OBUF       GET ADDRESS OF USERS BUFFER
      CLE,ELB        MAKE INTO BYTE ADDRESS
*
      LDA HRS        GET THE HOUR OF THE DAY
      JSB B2DEC      CONVERT IT AND STORE IT
      IFN
      LDA COLON
      SBT            PUT IN A COLON FOR A SEPARATOR
      XIF
      LDA MINS       GET THE MINUTES PAST THE HOUR
      JSB B2DEC      CONVERT
      IFN
      LDA COLON
      SBT
      LDA SECS       CONVERT THE SECONDS IF JULIS
      JSB B2DEC
      XIF
      LDA BLNK
      SBT            STORE TIME/DATE SEPARATOR
      XBX            SAVE THE POINTER
*
* TEST FOR LEAP YEAR AND COMPUTE DAY OF MONTH
*
      LDA YEAR       IS THIS A LEAP YEAR?
      AND D3         CHECK LEAST 2 BITS
      CLB
      CBY            (WILL NEED Y=0 LATER)
      SZA,RSS        IF 0, YEAR WAS EVENLY DIVISIBLE BY FOUR
      INB            SO MAKE FEBRUARY BE 29 DAYS INSTEAD OF
      ADB D28        THE NORMAL 28 DAYS
      STB MOTBL+1    AND STORE IT.
*
*   FOR THE PURIST, DIVIDING BY 4 TO TEST FOR LEAP YEARS IS
*   NOT A SUFFICIENT TEST, BUT THIS EASY TEST WILL NOT CAUSE AN
*   ERROR UNTIL 2100 A.D.
*
      LDA DAYM       NOW FIGURE OUT DAY OF MONTH
      STA TEMP       SET UP POINTER TO TABLE OF DAYS IN MO.
      LDA DAYS       GET DAY OF YEAR
      CMA,INA        MAKE IT NEGATIVE
*
MLP1  ISY            COUNT THE MONTHS IN Y REG.
      ISZ TEMP       BUMP MONTH POINTER
      ADA TEMP,I     SUBTRACT ONE MONTH'S DAYS
      SSA            WILL GO NEGATIVE IF ONE MONTH TOO FAR
      JMP MLP1
*
      CMA,INA        RESTORE REMAINDER
      ADA TEMP,I     NOW HAVE DAYS IN MONTH
      STA DAYS       SAVE TO PASS BACK IN A REG.
*
*  NOW PASS THE REMAINDER TO USER
*
      XBX            GET THE OUTPUT POINTER BACK
      JSB B2DEC      CONVERT DAYS TO ASCII
      CYA            GET MONTH NUMBER (1 TO 12) FROM Y
      STA TEMP
      ALS            MULTIPLY IT BY 3
      ADA TEMP       (3 TO 36)
      ADA DMOT       ADD BYTE ADDRESS ADJUSTED BY -3
      MBT D3         AND MOVE THE 3 LETTERS POINTED TO BY AREG
      LDA YEAR       GET THE YEAR
      ADA M1900      << THIS IS RESTRICTIVE TO 20TH CENTURY! >>
      JSB B2DEC      CONVERT IT AND STORE IN USERS BUFFER
*
      XBY            SET B=MONTH NUMBER
      LDA DAYS       SET A=DAY OF THE MONTH
      JMP RETRN,I    FINISHED
*
```

```
B2DEC NOP           BINARY TO DECIMAL CONVERSION
      XBX
      CLB           ASSUMING THE NUMBER IS OF FORM A=N*10+M,
      DIV D10       DIVIDE BY 10 TO SEPARATE N AND M, THEN
      XBX           ADD ASCII 0 TO MAKE THEM PRINTABLE
      ADA K60
      SBT           STORE THE M.S. DIGIT FROM AREG TO USER.
      CXA           GET THE L.S. DIGIT BACK
      ADA K60       MAKE IT ASCII
      SBT           AND STORE IT TOO.
      JMP B2DEC,I
*
M1900 DEC -1900     REVISE IN YEAR 2000, PLEASE.
D3    DEC 3
D10   DEC 10
D11   DEC 11
D28   DEC 28
K60   OCT 000060    ASCII ZERO
BLNK  ASC 1,        TWO ASCII BLANKS
      IFN
COLON ASC 1,::      TWO ASCII COLONS
      XIF
      SUP
DAYM  DEF MOTBL-1
MOTBL DEC 31,28,31,30,31,30,31,31,30,31,30,31
DMOT  DBR *-1       THREE LESS THAN THE BYTE ADDRESS OF *+1
      ASC 18,JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC
      BSS 0         REVEAL LENGTH OF PROGRAM
      END
```

KNOW THY COMPUTER

Alan Tibbetts/DSD

When you need to determine at run time which 21XX machine your program is running in, say for loading different microcode routines, you can use the following short section of code.

```
CLA,CCE         CLEAR A REGISTER AND SET E
ERA             MAKE A = 100000
CCB             MAKE B = 177777
OCT 100060      EXECUTE A "FUNNY" INSTRUCTION
NOP             NEEDED TO PREVENT BAD THINGS
```

When this is executed, you will find that the A and B register contents are different values, depending upon the machine that was used to execute the code.

```
MACHINE TYPE:  21MX-E  21MX-M  2100A(EAU)  2116C   2114B
A REGISTER:    100000  000000   000000     100000  100000
B REGISTER:    000000  040000   100000     177777  177777
```

Note that this program uses an "illegal" instruction, the OCT 100060 (sort of a LLS,ALS combination), which is not executed the same way in the different machines. Although the 2116 and 2114 are not micro-programmable, the information is included for comparison.

If you wish to find out what operating system is running the program, refer to "$OPSY — OPERATING SYSTEM TYPE" in the System 1000 Communicator, issue 13, pg. 26.

AUTO BOOT-UP FOR 21MXE COMPUTERS

Marlu Allan/DSD

The auto boot-up/RPL capability available with the HP 21MX E-Series computer has been enhanced to provide more flexibility. The new 21MX E-Series computer (2109B/2113B) contains a new power supply that is not dependent on a reset signal, so the full capability of auto boot-up/RPL can be realized. Soon all disc based HP 1000 Computer Systems sold will utilize auto boot-up/RPL. These systems will come up running when power is applied. The manual system boot which involved loading the S-Register with boot-up information (Loader ROM selection, loading device select code, channel selection on loading device), pressing PRESET, IBL and RUN is eliminated. All of the functions just described will be done automatically when power is applied to the system, or when a HLT instruction, 1060XX or 1070XX, is executed.

Three Loader ROMs are available for use with auto boot-up/RPL for loading from various devices. The 12992B (7905/20) Disc Loader ROM and the new 12992F (7900/01) Disc Loader ROM programs perform disc status checks before attempting to read from the disc. This allows the disc time to reach the READY state after power has been applied. The 12992E Flexible Disc Loader ROM can be used also.

The auto boot-up/RPL definition for the HP 21MX E-Series computers (2109A/2113A, A-Model) has been modified to provide more flexibility. The old definition is:

DESCRIPTOR BLOCK SWITCH	8	7	6	5	4	3	2	1	(closed=1)
CORRESPONDING S-REG BIT	15	14	10	9	8	7	6	0	
SWITCH FUNCTIONS	LOADER ROM		SELECT CODE					*RPL enable 0=not enabled 1=enabled	

The new definition of RPL for the HP 21MX E-Series computer (2109B/2113B, 'B-Model') is:

DESCRIPTOR BLOCK SWITCH	8	7	6	5	4	3	2	1	(closed=1)
CORRESPONDING S-REG BIT	15	14	10	9	8	7	6	0	
SWITCH FUNCTIONS	RPL enable & LOADER ROM		SELECT CODE					CHANNEL SELECT	

Descriptor block switch 8 acts as an RPL enable:

```
Switch 8 closed=1   RPL enabled
         open=0     RPL not enabled
```

As illustrated above, auto boot-up/RPL for the HP 21MX E-Series computer (2109B/2113B) allows selection of loader ROMs 10 or 11, select code, and the additional capability of channel selection.

Once the switches are configured, the computer will take appropriate action if auto boot-up/RPL is enabled. If it is enabled, then on power up or HLT(1060XX or 1070XX only) the microcode of the base set will store the configuration switches into the appropriate S-Register bits, jump to the IBL microcode routine, and jump to the RUN microcode routine which issues a RUN signal to the computer.

The auto boot-up/RPL capability allows the flexibility of booting from various devices and choosing the device channel from which to boot-up. Operator front panel interaction is kept to a minimum when using auto boot-up/RPL, since time consuming front panel operations are performed automatically under auto boot-up/RPL control.

NEW CONTRIBUTED PROGRAMS

DATA SYSTEMS LOCUS

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.

22682-18971 UPIT - RTE DEVICE LU UP

This program allows the RTE user to ensure that a range of device LU's are always up. It can be time-scheduled or scheduled once. UPIT could be called a line monitor. The routine is optimized for Real-Time and requires little resources of RTE. This routine is a must if you cannot afford operator intervention simply to make a device available.

22682-18971	PT	$10.00
22682-13371	Cass	$35.00

22682-10972 GCOPY RTE 7900/7905 TO MULTIPLE FLEXIBLE DISCS

GCOPY copies one 7900/7905, 96 sector/track, disc LU, to multiple flexible discs (60 sectors/tracks) in File Manager format. In other words, one 7900/7905 disc logical unit maps to multiple flexible disc cartridges. GCOPY starts at the top of the 7900/7905 directory and copies as many files as it can to the first flexible disc. It then asks for the next disc. This continues until all discs are copied. The user also has the option of skipping a disc or number of discs. By skipping through all the discs the first time, the user may determine the number of flexible discs needed for the copy. A directory list may be obtained for each flexible disc.

NOTE: Only one 7900/7905 directory track is allowed. No extents are allowed on source disc.

22682-10972	800	BPI MT	$40.00
22682-11972	1600	BPI MT	$40.00
22682-13372	Cass		$35.00

22682-18974 RTE READABLE PUNCH ROUTINE WITH SYSTEM DATE AND TIME

RDBLP - Does a look-up table conversion of an ASCII input buffer to a binary output buffer. The words in the output buffer are readable characters when punched on paper tape. Converts 64 character ASCII subset of upper case, numerals, and symbols.

JULIA/JULIS - Returns system time as a string:

FORMAT ---	JULIA		JULIS
1234	10DEC77	01:23:15	25JAN77

The day of the month is returned in the A-register and the month in the B-register.

TITLE - Puts a readable title, optionally containing the time/date, on a paper tape. The input is either from the string in the schedule call, or interactively from a terminal.

22682-18974	PT	$10.00

22682-10975 12555B D TO A RTE DRIVER

DVR55 is a RTE I, II, III & M Driver for the 12555B Digital-to-Analog Converter Interface Card. DVR55 processes four write requests and two control requests as listed below. Write Requests:

(1) Write to subchannel 0 - subfunction bit 6 set. First word in buffer is used as I * 10MS delay for outputting arrays in future write requests. "I" is the integer value of the first word in the buffer. I.E., CALL EXEC (2,-LU+100B,I,1)

(2) Write to subchannel 0 - no subfunction bits set. Output low half (1st 8 bits) to channel 1 of 12555B. Output high half (2nd 8 bits) to channel 2 of 12555B.

(3) Write to subchannel 1. Output low half of buffer word to channel 1 of 12555B. Channel 2 remains as previously programmed.

(4) Write to subchannel 2. Output low half of buffer word to channel 2 of 12555B. Channel 1 remains as previously programmed.

Control Requests:

(1) Control 0 - set both channels to 0 volts and clear the buffer rate output (EQT WD 14 to 0).

(2) Control subfunction bit 6 set - perform erase function and set both channels to 0 volts and clear buffer rate output.

DVR55 does not perform the refresh function and is not intended for graphics on a non-storage-type scope. DVR55 is intended for D to A operations (including stair step in 10MS increments), driving X-Y plotters or for graphics on storage-type scopes.

22682-10975	800	BPI MT	$40.00
22682-11975	1600	BPI MT	$40.00
22682-13375	Cass		$40.00

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	7/77	
02172-90005	Getting Started with Your HP 1000 Disc Based Computer System (for A computers)	4.00	6/77	
02172-90010	Getting Started with Your HP 1000 Disc Based Computer System (for B computers)	2.50	8/77	
02173-90007	Getting Started with Your HP 1000 System: Models 20 and 21	2.50	8/77	
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 Service 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 DVR 74 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	

RTE SYSTEMS MANUALS (Continued)

PART NUMBER	MANUAL TITLE	PRICE	DATE	UPDATE
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	
12732-90001	RTE Driver DVR33 Programming Manual	2.00	2/77	
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	
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	
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	5/77	9/77
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	
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	8/77
92060-90004	RTE-III Software System Programming and Operating Manual	12.00	7/77	8/77
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	
92060-90012	RTE: A Guide for New Users	6.50	7/76	
92060-90013	Batch-Spool Monitor Reference Manual	9.50	3/77	
92060-90014	RTE Interactive Editor Reference Manual	6.00	5/77	
92060-90017	RTE Utility Programs	3.00	3/77	
92060-90020	RTE On-Line Generator	15.00	7/77	
92064-90002	RTE-M Programmer's Reference Manual	14.00	3/77	7/77
92064-90003	RTE-M System Generation Reference Manual	7.50	3/77	7/77
92064-90004	RTE-M Editor Reference Manual	6.00	1/77	3/77
92064-90007	RTE-M Pocket Guide	4.50	8/77*N	
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	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 M-Series Computer Reference Manual	$ 5.50	6/76	7/76
02108-90006	HP 21MX M-Series Computer Installation and Service Manual	10.00	7/76	
02108-90004	HP 21MX M-Series Computer Operators Manual	5.00	7/76	
02108-90017	21MX M-Series Computer Engineering and Reference Documentation	125.00	5/77*R	
02108-90027	21MX K-Series Computer Engineering and Reference Documentation	100.00	5/77*R	
02109-90001	HP 21MX E-Series Computer Operating and Reference Manual	8.00	7/77*R	
02109-90002	HP 21MX E-Series Computer Installation and Service Manual	15.00	8/76	9/77
02109-90006	HP 21MX M- and E-Series Computer I/O Interfacing Guide	7.00	7/77*R	
02109-90014	21MX E-Series Computer HP 2109B and HP 2113B Operating and Reference Manual	8.00	8/77*N	
02109-90015	21MX E-Series Computer HP 2109B and HP 2113B Installation and Service Manual	15.00	8/77*N	9/77
12732-90005	HP 12732A/12733A Flexible Disc Subsystem Operating and Service Manual	5.50	8/77*R	
12979-90006	HP 12979A I/O Extender Installation and Service Manual	15.00	6/77*R	9/77
12979-90007	HP 12979A I/O Extender Operating and Reference Manual	5.00	12/75	9/77
12979-90014	HP 12979B Input/Output Extender Operating and Reference Manual	2.00	8/77*N	
12979-90016	HP 12979B Input/Output Extender Installation and Service Manual	12.00	8/77*N	8/77
12990-90003	HP 12990A Memory Extender Installation and Service Manual	5.50	4/76	8/76
5950-3765	21MX E-Series Computer Technical Reference Manual	3.50	6/77*N	

LANGUAGE MANUAL

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

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 1740 (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
92067-16004	!S4LP7	24K SIO LINE PRINTER DRIVER	92001-13305	1538
92061-16021	%DVR15	RTE 7261A DRIVER	92062-13304	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	1726
1297x-16004	!S4MT1	24K SIO MAG.TAPE DRIVER	92001-13305	1550
29074-60001	%DVR30	RTE FIXED HEAD DISC DRIVER	92062-13305	C
29808-60001	%CAL10	CAL. PLOTTER DRIVER	92062-13302	B
29810-60001	%CALIB	CAL. PLOTTER LIBRARY	92062-13302	C
29875-60001	%1FTN	FORTRAN MAIN CONTROL	92060-13306	E
29875-60002	%2FTN	FORTRAN PASS 1	92060-13306	E
29875-60003	%3FTN	FORTRAN PASS 2	92060-13306	E
29875-60004	%4FTN	FORTRAN PASS 3	92060-13306	E
29875-60005	%5FTN	FORTRAN PASS 4	92060-13306	E
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	1740*
24998-16001	%RLIB2	RTE/DOS LIBRARY PART 2	92060-13302	1740*
24998-16002	%FF4.N	FORTRAN IV FORMATTER	92060-13303	1726
25117-60459	%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	1740*
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	!S4MT2	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	%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	B
92001-16005	%SYLIB	RTE SYSTEM LIBRARY	92060-13301	1740*
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	1740*

(Continued)

SOFTWARE MODULE NUMBERS: 92001B LEVEL 1740 (RTE II)

PAPER TAPE	MODULE	DESCRIPTION	CARTRIDGE	DATE CODE
92001-16013	%2GN00	RTE-II 7900 OFF-LINE GEN.	92001-13303	1631
92001-16014	%AUTOR	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	1740*
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13302	1740*
92001-16029	%SCMD2	RTE-II COMMAND PROGRAM	92001-13301	1710
92001-16030	%WHZT2	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
92001-16035	%DVA05	RTE DRIVER 264X MODEM	92062-13302	1740*
92001-16014	&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 ANS* 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	%DFKLB	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	1740*
92060-18047	&PKDIS	PACK DISC TRANSFER FILE	92001-13302	1631
92064-16086	%MSAFD	FLEXIBLE DISC BACKUP UTILITY	92060-13309	1740*
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	1631

SOFTWARE MODULE NUMBERS: 92060B LEVEL 1740 (RTE III)

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

PAPER TAPE	MODULE	DESCRIPTION	CARTRIDGE	DATE CODE
92607-16004	!S4LP7	24K SIO LINE PRINTER DRIVER	92001-13305	1538
29601-16021	%DVR15	RTE 7261A DRIVER	92062-13304	A
12732-16001	%DVR33	FLEXIBLE DISC DRIVER	92062-13304	1726
1297?-16004	!S4MT1	24K SIO MAG. TAPE DRIVER	92001-13305	1550
20747-60001	%DVR30	RTE FIXED HEAD DISC DRIVER	92062-13305	C
20808-60001	%CAL1B	CAL. PLOTTER DRIVER	92062-13302	B
20510-60001	%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	%ALGL1	RTE/DOS ALGOL PART 2	92060-13305	C
24153-60001	%FF.N	RTE/DOS FORMATTER	92060-13303	C
24316-60001	%DECAR	DOSM ST ARITH PK	92060-13303	A
24998-16001	%RLIB1	RTE/DOS LIBRARY PART 1	92060-13302	1740*
24998-16002	%RLIB2	RTE/DOS LIBRARY PART 2	92060-13302	1740*
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	1740*
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	!S4MT2	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	%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
92001-16002	%LDR2	RTE LOADER	92001-13301	1726
92001-16003	%MTM	MULT. TERMINAL MONITOR	92001-13301	B
92001-16004	%2DP43	POWER FAILURE DRIVER	92001-13301	1633
92001-16005	%SYLIB	RTE SYSTEM LIBRARY	92001-13301	1740*
92001-16012	%CR2SY	CORE RESIDENT OPERATING SYS.	92001-13301	1740*

(Continued)

SOFTWARE MODULE NUMBERS: 92060B LEVEL 1740 (RTE III)

PAPER TAPE	MODULE	DESCRIPTION	CARTRIDGE	DATE CODE
92001-16028	%DV05	RTE 2640A DRIVER	92062-13302	1740*
92001-16035	%DVA05	RTE DRIVER 264X MODEM	92062-13302	1740*
92001-16014	%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	%3SP01	RTE-III SPOOL MONITOR PART 1	92060-13313	1631
92060-12001	%3SP02	RTE-III SPOOL MONITOR PART 2	92060-13313	1631
92060-12003	%CR3SY	MEMORY RESIDENT SYSTEM	92060-13301	1740*
92060-12004	%ASMB	RTE ASSEMBLER	92060-13304	1639
92060-12005	%CLIB	RTE COMPILER LIBRARY	92060-13315	1726
92060-16001	%3DF43	POWER FAILURE DRIVER	92060-13301	1633
92060-16004	%LDR3	RTE-III LOADER	92060-13301	1726
92060-16006	%WHT13	RTE-III WHZAT PROGRAM	92060-13310	1726
92060-16028	%XREF	CROSS REFERENCE	92060-13304	A
92060-16029	13GN00	7900 RTE-III GENERATOR	92060-13311	1631
92060-16031	%DVR32	RTE 7905A DISC DRIVER	92062-13305	A
92060-16032	13GN05	7905 RTE-III GENERATOR	92060-13311	1631
92060-16035	%SPVMP	SPVMP	92060-13301	A
92060-16036	%SCMD3	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 (RSTOR)	92060-13309	1704
92060-16041	%VERFY	DISC VERIFY PROGRAM	92060-13309	1704
92060-16042	%COPY	DISC COPY PROGRAM	92060-13309	1704
92060-16043	%DBKLB	DISK BACK 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	92060-13316	1726
92060-16098	%4FTN4	FORTRAN IV SEGMENT 4	92060-13316	1726
92060-16046	%UPDAT	UPDATE TRANSFER FILE	92060-13310	1740*
92060-16047	%PKDIS	PACK DISK TRANSFER FILE	92060-13310	1631
92060-16050	%AN3F0	RTE-III 7900 GFATHER ANSW FILE	92060-13314	1726
92060-16051	%AN3F5	RTE-III 7905/20 GFATHER ANS FILE	92060-13314	1726
92064-16086	%MSAFD	FLEXIBLE DISC BACKUP UTILITY	92060-13309	1740*
92202-16001	%DVR23	RTE 7970 9T. MAG. TAPE DRIVER	92062-13304	A
92202-16002	%2DV47	RTE 9895A DRIVER WITHOUT DMS	92062-13302	1726
92202-16003	%3DV47	RTE 9895A DRIVER WITH DMS	92062-13302	1643

SOFTWARE MODULE NUMBERS: 92062A LEVEL 1740 (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
29026-60002	%DVR12	RTE 2767A DRIVER	92062-13303	A
29029-60001	%DVR00	RTE TTY/PUNCH/PHOTO READER	92062-13302	1740*
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	9120ØA DRIVER	92062-13303	1648
91200-16002	%TVLIB	91200A VIDEO MONITOR LIBRARY	92062-13303	1648
91200-16004	%TVVER	9120ØA 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	1740*
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13302	1740*
92001-16035	%DVA05	RTE DRIVER 264X MODEM	92062-13302	1740*
92060-16031	%DVR32	RTE 7905A DISC DRIVER	92062-13305	A
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 1740 (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 1740 (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
20806-60001	%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	92064-13402	C
24153-60001	%FF,N	RTE/DOS FORTRAN FORMATTER	92060-13303	92064-13401	C
24306-60001	%DECAR	DOSM STRING ARITH PK	92060-13303		A
24998-16001	%RLIB1	RTE/DOS LIBRARY	92060-13302	92064-13401	1740*
24998-16001	%RLIB1	RTE/DOS LIBRARY	92060-13302	92064-13402	1740*
24998-16001	%RLIB2	RTE/DOS LIBRARY	92060-13302	92064-13402	1740*
24998-16001	%RLIB2	RTE/DOS LIBRARY	92060-13302	92064-13401	1740*
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	1740*
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	1740*
92001-16028	%0DV05	RTE 2640A DRIVER	92062-13302	92064-13401	1740*
92001-16035	%DVA05	RTE DRIVER 264X MODEM	92062-13302	92064-13401	1740*
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

(Continued)
SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1740 (RTE-M)

PAPER TAPE	MODULE	DESCRIPTION	CARTRIDGE	FLEXIBLE DISC	DATE CODE
92064-16006	%MMP	MI SCHEDULING OPTION	92064-13301	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MII)	92064-13302	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MIII)	92064-13303	92064-13401	1650
92064-16008	%MTI	TIMER OPTION (MI)	92064-13301	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MIII)	92064-13303	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MII)	92064-13302	92064-13401	1650
92064-16009	%MTS	TIME SCHEDULING OPTION (MI)	92064-13301	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MIII)	92064-13303	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MII)	92064-13302	92064-13401	1650
92064-16010	%MOP	OPERATOR COMMAND OPTION (MI)	92064-13301	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 (MI)	92064-13301	92064-13401	1650
92064-16013	%MDMLB	DUMMY LIBRARY (MIII)	92064-13303	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 (GEN DISC)	92064-13305	92064-13401	1726
92064-16023	%RTMLD	RELOCATING LOADER (APP DISC)	92064-13305	92064-13402	1726
92064-16024	%RTMSC	LOADER SUB CONTROL (APP DISC)	92064-13305	92064-13402	1726
92064-16024	%RTMSC	LOADER SUB CONTROL (GEN DISC)	92064-13305	92064-13401	1726
92064-16025	%MEDIT	EDITOR		92064-13402	1703
92064-16026	%MASM6	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 MNGR (MIII)	92064-13303	92064-13401	1650
92064-16031	%MRN	RESOURCE NUMBER MANAGER (MII)	92064-13302	92064-13401	1650
92064-16032	%ONMTM	MULTI TERMINAL MONITOR (APP D)	92064-13305	92064-13402	1650
92064-16032	%ONMTM	MULTI TERMINAL MONITOR (GEN D)	92064-13305	92064-13401	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

(Continued)
SOFTWARE MODULE NUMBERS: 92064A OPTIONS 20 & 40 LEVEL 1740 (RTE-M)

PAPER TAPE	MODULE	DESCRIPTION	CARTRIDGE	FLEXIBLE DISC	DATE CODE
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 (APP DISC)		92064-13402	1650
92064-16056	%DRF	F DISC DIRECT PROG (GEN DISC)		92064-13401	1650
92064-16057	%TBLFP	FLEXIBLE DISC DIRECT TABLES		92064-13401	1709
92064-16060	%DRF1	F DISC DIRECTORY SUB (APP D)		92064-13402	1650
92064-16060	%DRF1	F DISC DIRECTORY SUB (GEN D)		92064-13401	1650
92064-16075	1MFGEN	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
92064-16086	%MSAFD	FLEXIBLE DISC BACKUP UTILITY	92060-13309	92064-13402	1740*
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		•		Nov 30								
	3 days	300											
22940A	2100 Maint.			Nov 7 Dec 5									
	10 days	$1000											
22941A	21MX Maint.			Nov 28					Dec 5				
	5 days	500											
22942A	7900 Maint.			Nov 28					Nov 28				
	5 days	500											
22943A	7970B Maint.					Nov 14							
	5 days	600											
22944A	7970E Maint.					Nov 7							
	5 days	600											
22945A	7905 Maint.			Nov 7 Dec 5 Dec 12					Nov 14				
	5 days	500											
22950A	2100 Ser. Assm.		Nov 7 Dec 5		Nov 7 Dec 5		Nov 21 Jan 30		Dec 12	Nov 14 Jan 23		Nov 28	
	5 days	500											
22965B	RTE-II/III		Nov 7 Dec 5		Nov 7 Nov 14 Nov 28 Dec 5 Dec 12 Dec 19		Nov 7 Nov 28 Dec 5 Jan 9 Jan 16	Nov 14 Nov 21	Nov 21 Dec 5 Jan 9 Dec 5	Nov 28 Dec 19		Dec 5 Dec 12	
	10 days	1000											
	(Course includes RTE-II/III operating system, batch spool monitor and file manager.)												
22969A	Distr. Sys.		Nov 28						Nov 28				
	5 days	500											
22977A	Image/DBMS 1000		Dec 12						Jan 16	Jan 9			
	5 days	500											
22980B	HPIB Minicomputer Environment		Nov 28						Nov 7				
	4 days	400											

TITLE TRAINING COURSE RATES AND CENTER LOCATION

Course Number	Length	Price	Cupertino	Sunnyvale	Rockville	Boise	Boblingen	Winnersh	Grenoble	Milan	*** Madrid	Stockholm	*** Amsterdam/ Brus.
22983A	21MX E-Micro-programming		Nov 14, Dec 12										
	5 days	500											
22984A	7920 Maint.			Nov 14									
	5 days	500											
22985A	RTE-M		Nov 7 Dec 5		Nov 14				Dec 12				
	5 days	500											

*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.

***We have not yet received dates for Madrid or Amsterdam/Brus.

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 FOR LOCUS CONTRIBUTED SOFTWARE

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.

D00 — Documentation

Not all options are available for all programs.

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

For example:

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

Only those options listed in each abstract are available.

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

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

HEWLETT-PACKARD
LOCUS CONTRIBUTED SOFTWARE CATALOG
DIRECT MAIL ORDER FORM

Please Print:

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Country _____

☐ HP Employee Account Number _____ Location Code _____

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

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

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

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

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

NOT TO BE USED FOR ORDERING COMMUNICATOR SUBSCRIPTIONS

HEWLETT hp PACKARD

CORPORATE PARTS CENTER

Direct Mail
Parts and Supplies Order Form

SHIP TO:

NAME _____

COMPANY _____ CUSTOMER REFERENCE # _____

STREET _____ TAXABLE ? _____

CITY _____ STATE _____ ZIP CODE _____

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

Special Instructions

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

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

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

When completed, please mail this form with payment to:

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

Phone: (415) 968-9200

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