SORT/250 Programming Manual

Manual Part No. 45251-90024

Hewlett-Packard
19447 Pruneridge Avenue, Cupertino, California 95014

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revised date at the bottom of the page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

August 1978 ... **FIRST EDITION**
June 1979...**SECOND EDITION:** Revised pages: 2-3, 2-6 thru 2-9, 3-7, 3-14, B-1, C-5, C-6, D-1 thru D-3

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Table of Contents

Chapter 1: Introduction

Chapter 2: SORT Statements and Functions

Chapter 3: Program Examples

Chapter 4: Programming Considerations

Appendix C: Determining Workfile Size

CHAPTER 1
Introduction

This manual describes the SORT/250 software available with the HP250 Business System. Chapter 1 presents a brief overview of SORT/250 terms and concepts. Chapter 2 describes the syntax of the various statements and functions. Chapter 3 lists sample programs using SORT/250. Chapter 4 covers optimization techniques.

This manual is intended for the programmer who is familiar with both the HP250 BASIC Programming Manual and the IMAGE/250 Programming Manual.

What is SORT/250?

SORT/250 is a collection of BASIC statements and functions to facilitate retrieving information from an IMAGE/250 data base. Statements are provided to allow accessing data in sorted order, and for selecting subsets of the total information available.

In addition, SORT/250 provides for simulating structures more complex than the two-level networking supported by IMAGE/250. SORT/250 enables the program to access a data base in a hierarchical fashion. Simple data sets (a degenerate hierachy) can also be handled as well as certain non-hierarchical structures.

Specifying Data Base Structure

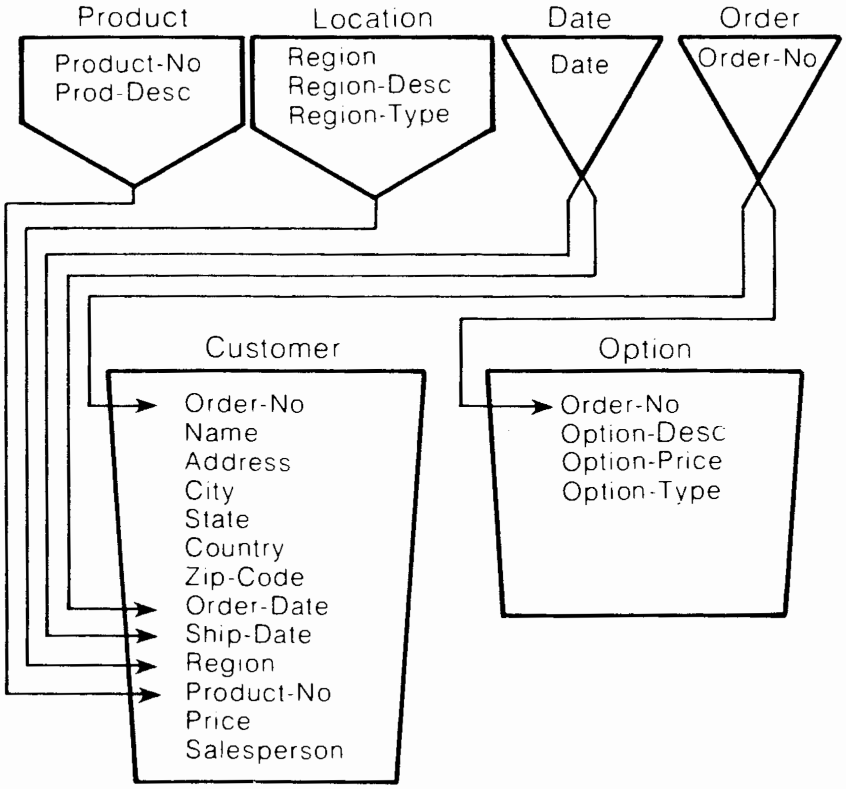
Before beginning any actual data base access via SORT/250, it is necessary to specify the structure of that portion of the data base to be used. This structure is specified as a list of set names, optionally separated by information concerning their interrelationship. This list is called the **thread**. The thread specification describes the hierarchical (or other) structure on which SORT/250 statements operate.

SORT/250 operations are used to extract information according to the thread specification. This information is in the form of record pointers which the program uses in direct-mode DBGETs to obtain the actual information from the data base. The thread may contain from one to ten sets, depending on the particular application.

The next diagram shows the example Sales Analysis Data Base. Some example reports which can be obtained are:

1. A list of all orders.
2. A list of products along with the orders placed for that product.
3. A list of products and the orders, as above, but including the options contained with each order.

To produce report 1, only the CUSTOMER data set is involved. The thread list for such a structure would consist only of CUSTOMER. Report 2 involves the data sets PRODUCT and CUSTOMER, while 3 involves the sets PRODUCT, CUSTOMER and OPTION. A further discussion of how to generate these reports is in this chapter, along with the program to generate report 1. Chapter 3 gives complete sample programs for reports 2 and 3.



Sales Analysis Data Base

Schema listing for this data base is in Appendix A.

Putting Data into Sorted Order

The SORT BY statement allows a sort to be determined by up to ten data items from any data set in the thread. If an ordering of two elements cannot be determined on the basis of the first field, the second, the third, and so on, will be compared until an ordering can be found. If no ordering is found, the pointers into the data base are compared to determine order. Additionally, sort direction is specifiable on each sort field on an individual basis. Any field may be suffixed by the keyword DES to cause it to sort in descending order rather than ascending.

Here is the same program shown earlier, but with some additional statements filled in:

In this example, lines 80 thru 100 are used to create a file, ASSIGN it to a file position and convert it into a workfile. (Note that the file is still of type DATA.) Line 120 produces pointers so the data can be accessed in sorted order. Line 170 reads the pointer into a BASIC variable so it can be used in the direct mode DBGET in line 180.

One additional function has been introduced in this example, the WFLEN function used in line 130. This function returns the number of pointers in the workfile. It has as an argument, the file number of the workfile, since more than one workfile may be in use at a given time. Notice that the program creates and purges the workfile each time the program is run. If disc space is available, program execution time can be decreased by deleting lines 80 and 220, which allows the file to remain on the disc.

```
10    DIM Buf$[170],B$[5],Order_no$[10],Name$[30]
20    INTEGER S(9)     ! Ten-element status array.
30    B$="  SAD"
40    DBOPEN (B$,"MANAGER",3,S(*)) ! Open the database.
50    DBASE IS B$
60    IN DATA SET "CUSTOMER" USE Order_no$,Name$
70    ! Now set up a workfile with CUSTOMER as the thread.
80    FCREATE "XYZ",200
90    ASSIGN "XYZ" TO #1
100   WORKFILE IS #1;THREAD IS "CUSTOMER"
110   ! Sort the orders by order number.
120   SORT BY Order_no$
130   Entry_count=WFLEN(1)    ! WFLEN returns no. of pointe
rs in file.
140   PRINT " ORDER NUMBER";TAB(30);"CUSTOMER NAME";LIN(2)
150   FOR I=1 TO Entry_count
160     ! Read the record pointer into Rec_no.
170     READ #1;Rec_no
180     DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,Rec_no)
190     PRINT Order_no$;TAB(30);Name$
200   NEXT I
210   DBCLOSE (B$,"",1,S(*))
220   PURGE "XYZ"
230   END
```

Selecting Data

Often times only a small portion of the total available space is of interest for processing purposes. SORT/250 provides the FIND statement to select only those entries in the hierarchy which are pertinent. This selection can involve data available at any level of the hierarchy and may use an arbitrarily-complex selection criteria involving any function available in a BASIC expression.

When a FIND is executed, pointers to some subset of the records in the hierarchy are put in the workfile. Only the pointers for records which meet the selection criteria are put in the workfile. If there are already pointers in the workfile from executing previous FINDs (or SORTs), then the subset described by these pointers is used in successive FINDs and SORTs, rather than all the information present in the data base.

Suppose, in the above example, it was desirable to list only the orders for ABC Company. This can be accomplished by inserting a FIND statement somewhere between line 100 and line 130 to select only those customers. Thus a report for just ABC Company could be produced by adding:

```
115  FIND TRIM$(Name$)="ABC Company"
```

This line could have followed the SORT BY in line 120 since executing a FIND does not change the order produced by the last SORT BY.

Note the use of TRIM$. This is necessary because FIND works like a direct-mode DBGET. The unpacking procedure performed by IN DATA SET will leave any trailing blanks on the string.

Suppose, now, that an additional restriction needs to be put on the set of orders in the report. The report should contain only those orders from ABC Company and only if there is a "2" somewhere in the order number. This can be accomplished in either of two ways. The first method is to add another FIND statement specifying the additional restriction between lines 100 and 130. The second method is to change line 115. The first method might produce a line like[1]:

```
125  FIND POS(Order_no$,"2")<>0
```

A more efficient way is the second method. The fewer FIND statements executed the better, since each data entry need be examined only once. (This is the usual case. More details on the best way to optimize FINDs are presented in Chapter 4.) This method might have produced a replacement for line 115 such as:

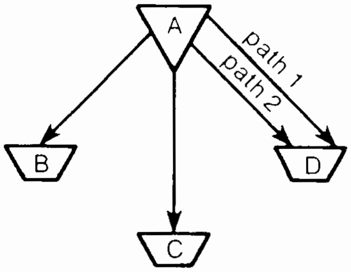
```
115  FIND (TRIM$(Name$)="ABC Company") AND (POS(Order_no$,"2")<>0)
```

[1] Note that now one of the FINDs is before the SORT BY and one is after. Both could also appear before or both after the SORT BY.

Specifying Complex Data Base Structures

As indicated earlier, it is sometimes useful to sort or find records spread over several data sets when those data sets logically represent a hierarchy. The thread parameter on the workfile statement allows this to be done. The thread is basically a list of the sets in the order they occur in the hierarchy.

The following figures show one master with three detail sets linked to it.



A Multiple Two-Level IMAGE Structure

{A} or {B} or {C} or {D}

{A,B} or {B,A}
{A,C} or {C,A}

{B,A,C} or {C,A,B}

Threads Defined

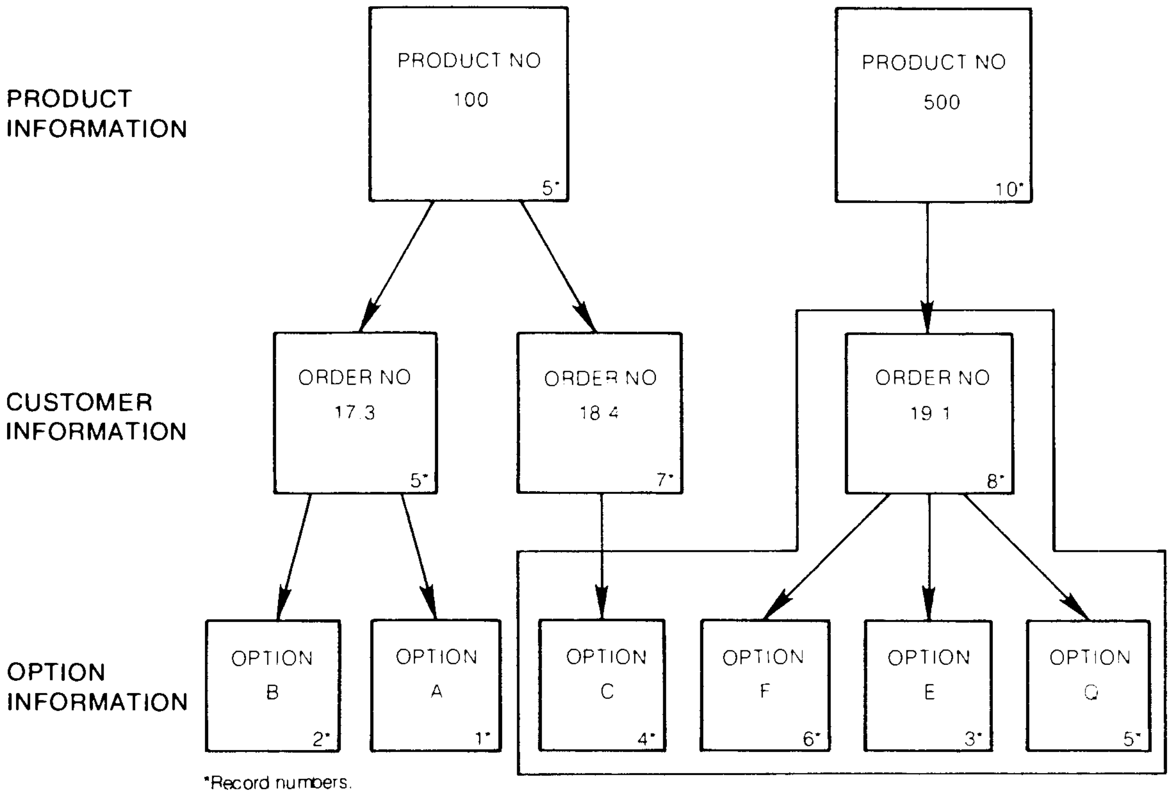
Notice that detail data set D has two data paths to the same master. In this case, linking set A to set D is ambiguous. To resolve this ambiguity, it is necessary to specify which path is involved. Adding this capability to the thread specification allows the description of the following additional threads:

Additional Threads

{A (via path 1) D (via path 2) A}
{A (via path 2) D (via path 1) A}
{D (via path 1) A (via path 2) D}
{D (via path 2) A (via path 1) D}
{C,A (via path 1) D (via path 2) A,B}
{B,A (via path 2) D (via path 1 A,C}
etc.

Remember that although these threads can be defined, they may not make any sense! It is the programmer's responsibility to determine the sense of a thread.

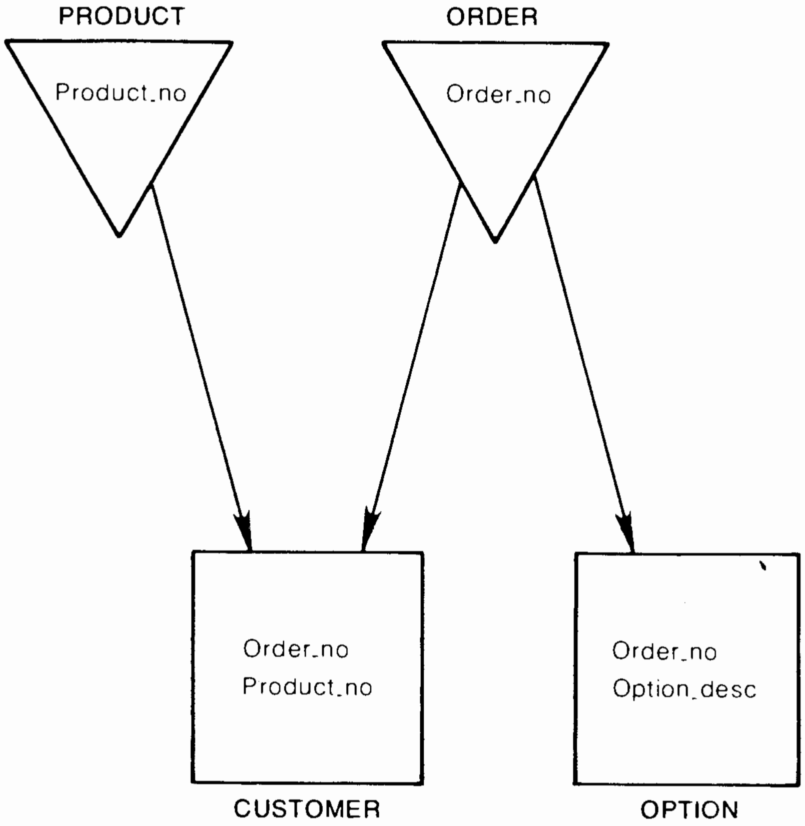
As another example, refer to the three reports introduced on page 1-2. Generating report 2 involves using two sets. The thread that describes this hieraracby is specified as a list of PRODUCT and CUSTOMER. Report 3 involves three sets (PRODUCT, CUSTOMER and OPTION). The structures involved in all these reports are hierarchical in nature. In report 2, for example, the PRODUCT data set is higher in the hierarchy than CUSTOMER. Report 3 is an example of a three-level hierarchy. The next figure shows how the hierarchy for report 3 is organized.



*Record numbers.

Sample Three-level Heirarchy

Unlike report 2 where there is a direct connection between PRODUCT and CUSTOMER, there is no connection between CUSTOMER and OPTION. This is why the ORDER master data set exists. The thread necessary for accessing this three-level hierarchy consists of four sets which are specified in the order PRODUCT, CUSTOMER, ORDER and OPTION. See the next figure.



Simulation of a Three-level Hierarchy

A sample output for report 3 is shown next. Notice that information is obtained from the product data set (product number and description), as well as from each of the other sets. Graphically, this information is organized as shown on page 1-8. The numbers in the corner of the boxes correspond to the records where the information is stored in the data base. Entries for the ORDER detail are not shown, since the ORDER set contains no information pertinent to producing the report.

```
                         OUTSTANDING ORDERS LIST

PRODUCT NO.          ORDER NO.   CUSTOMER NAME      OPTIONS        PRICE
--------------------------------------------------------------------------
100(STD BICYCLE)     17.3        XYZ Company          A            10.25
                                                      B            20.31
                                                                --------
                                                                   30.56

                     18.4        XYZ Company          C            30.97
                                                                --------
                                                                   30.97
                                                                =======
                                 TOTAL 100 ORDERS:                 61.53

500(5-SPEED)         19.1        ABC Company          E            123.05
                                                      F            100.1
                                                      Q            1.23
                                                                --------
                                                                   224.38
                                                                =======
                                 TOTAL 500 ORDERS:                 224.38

                                 TOTAL ORDERS:                     285.91
                                                                =======
```

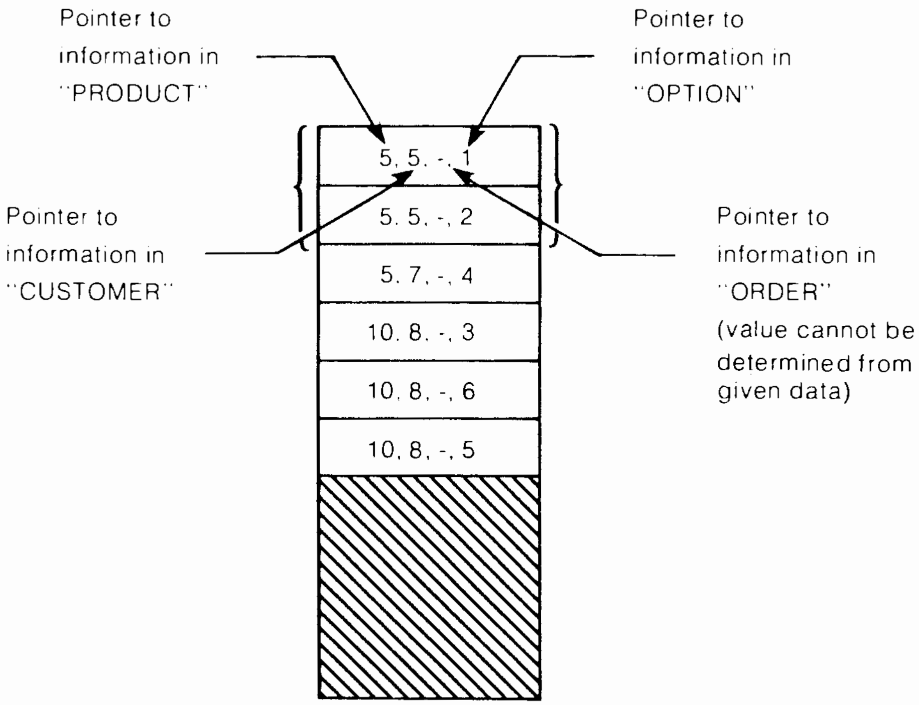
To produce report 3, it is necessary to extract this information from the data base (record numbers from figure on page 1-8):

Set Name	Record to Read	Action to Take
Product	5	Print header for product.
Customer	5	Print header for order.
Option	1	Print first option.
Option	2	Print last option and total.
Customer	7	Print header for new order.
Option	4	Print option and totals.
Product	10	Print header for new product.
Customer	8	Print header for order.
Option	3	Print first option.
Option	6	Print second option.
Option	5	Print last option and totals.

The numbers stored in the workfile, however, always contain one record from each set. Thus, the first record will contain the three pointers in color above and the pointer to the ORDER set.

The subsequent record is the same except that the pointer for the option set is changed to 2. The next figure shows the pointers as they are stored in the workfile.

Note that one pointer for each set is always stored. If a record at one level of the hierarchy has no records associated with it at the next lower level, there is no way to store a record of pointers in the workfile pertaining to that record. In particular, if the records circled in the figure on page 1-8 are deleted, product 500 has no order associated with it and order 18.4 has no associated options. The workfile would then have only two records corresponding to the bracketed records in the next figure. Further, if the options on order number 17.3 were deleted, FIND or SORT would return an empty workfile.


Pointer to information in "PRODUCT"
Pointer to information in "OPTION"
Pointer to information in "CUSTOMER"
Pointer to information in "ORDER" (value cannot be determined from given data)
5, 5, -, 1
5, 5, -, 2
5, 7, -, 4
10, 8, -, 3
10, 8, -, 6
10, 8, -, 5

Contents of Workfile after Sorting

The program to produce the outstanding order list is fairly complex, as shown in Chapter 3. However, the skeleton for the program is shown next. This skeleton reads four pointers from the workfile even though the third pointer (to the automatic master set ORDER) is not used. Also, note that this skeleton repeatedly reads records from the PRODUCT and CUSTOMER data set even though it may be reading the same record as on the previous pass through the loop. For clarity's sake, the code to optimize out the extra reads is not shown.

```
ASSIGN "XYZ" TO #1
WORKFILE IS #1:THREAD IS "PRODUCT","CUSTOMER","ORDER","OPTION"
•
•
•

IN DATA SET "CUSTOMER" USE ALL
IN DATA SET "OPTION" USE SKP 1,Option_desc$,PO
•
•
•

SORT BY Product_no,Order_no$,Option_desc$
•
•
•

FOR L=1 TO WFLEN(1)
  READ #1;R1,R2,R3,R4
  DBGET (Base$,"PRODUCT",4,S(*),"@",Buf$,R1)
  DBGET (Base$,"CUSTOMER",4,S(*),"@",Buf$,R2)
  DBGET (Base$,"OPTION",4,S(*),"@",Buf$,R4)
•
•
•
NEXT L
```

CHAPTER 2

SORT Statements and Functions

Introduction

This chapter describes the syntax needed to use SORT/250 software. The statements and functions provided with SORT/250 are:

WORKFILE IS # A statement specifying the hierarchical structure (thread) of the data sets to be sorted, the work space for sorting, and the workfile itself.

SORT BY A statement specifying the order in which data is to be sorted.

FIND A statement used to select a subset of record pointers from the data base or the current workfile.

WFLEN A function returning the number of logical records in the workfile.

Two IMAGE/250 statements, DBASE IS and IN DATA SET, are used to define the data base and data sets before unpacking data entries with SORT/250.

In addition, many BASIC file storage operations (PRINT #, READ #, REC, etc.) are used in conjunction with SORT/250 workfiles. Because of the workfile structure, these operations may work differently with SORT/250 than as described in the BASIC Programming Manual. These differences are covered near the end of the chapter.

Syntax Conventions

The statements in this manual use the same syntax conventions as in the BASIC Programming Manual.

DOT MATRIX All keywords and characters in dot matrix must appear as shown.

... An ellipsis indicated that the previous parameter can be repeated.

[ ] All parameters in brackets are optional. If there are brackets within brackets, the parameter within the inner bracket may only be specified if the parameter in the outer bracket is spcified. Parameters may also be stacked in brackets. For example: $\begin{bmatrix}A\\B\end{bmatrix}$ A or B or neither may be selected.

{ } One parameter must be selected from those stacked within braces. For example: $\begin{Bmatrix}A\\B\\C\end{Bmatrix}$ A or B or C must be selected.

The WORKFILE IS # Statement

The WORKFILE IS # statement describes the hierarchical structure on which FIND and SORT will operate, where the scratch area is for SORT, and where the results of executing a FIND or SORT are stored.

```
WORKFILE IS #file number[;THREAD IS [set id[LINK link];...]set id]
                                            [;path id ]
                                    \_________________________________/
                                                thread list
                                          (up to 10 sets allowed)
```

The parameters are:

file number A numeric expression having an integer value from 1 through 10, and used to identify a file previously defined by an ASSIGN statement.

set id A numeric or string expression used to identify a data set. If numeric, this parameter references a data set number for the current data base (specified in the last DBASE IS statement). If a string, this parameter references a data set name for the current data base.

path id — A numeric expression having an integer value from 1 through 8. This expression selects which data path to use between the first data set specified (set id) and the next in the thread list. It is needed only when more than one path exists between two sets being linked in the thread. If only one path exists for the data set specified, it is not necessary to list the path id parameter.

link — A BASIC variable which is currently linked via an IN DATA SET statement to an item found in the detail data set to which it is attached. The variable must match in type and length the search item in the master data set which follows in the thread list. If the variable refers to a sub-item, it may only be the first sub-item.

NOTE

The path or link parameters cannot be specified on the last data set in the thread list, since these operations specify a relationship between the set to which it is attached and the next set listed in the thread.

Some examples of the WORKFILE IS # statement are -

```
WORKFILE IS #1;THREAD IS"CUSTOMER"
WORKFILE IS #X+3;THREAD IS"CUSTOMER":2,"DATE"
WORKFILE IS #8;THREAD IS"CUSTOMER":2,"DATE"
```

Up to 10 data sets can be specified for any thread list. The number of sets in the list is referred to as the **thread length**. Each set must be related to the sets on either side of it (or one side if it is at the end of the thread) by a path in the data base (or a **synthetic path** using the LINK option). This defines the hierarchical structure, with the leftmost set in the thread list usually being the highest (usually the least commonly occuring) in the hierarchy. Successful execution of WORKFILE IS # converts the file into a workfile. To convert a file to a workfile, the file must be ASSIGNed in exclusive mode. The file remains a workfile until either another file is assigned in its place (same file number) or it is de-assigned. Closing the data base to which the workfile pertains automatically de-assigns the workfile.

A workfile uses 214 bytes of user memory if it was un-buffered at the time it was converted to a workfile. If it was BUFFERed, no additional memory space is used, since BUFFER # uses 270 bytes of user memory. When the workfile is de-assigned, the memory space is returned.

The workfile is used to store all pointers generated by FIND and SORT BY operations. Initially, the workfile contains no pointers, so any attempt to access them (via READ #) will result in an error. The REC function returns 0 to indicate this null state. Pointers can be put in the workfile by executing SORT BY, FIND, FIND ALL or PRINT #.

The workfile is composed of logical records whose lengths in bytes are twice the thread length. Thus, a two-byte pointer is stored for each set in the thread in any given logical record. Pointers may range in value from one to the capacity of the set to which they pertain. (The first pointer in the record corresponds to the first set in the thread, the second pointer corresponds to the second set, and so on.) The workfile also makes use of any additional space on the last file sector, space normally inaccessible when the file is not a workfile.

In the case where more than one path connects two adjacent sets in the thread, it is necessary to specify which path is to be used. This is done by suffixing the first of the sets with a ":" and following that with a path id. The path id for a particular path is determined by using the schema listing. To find the path with path id **n**, for example, scan the detail for the **n**th occurrence of the master set name. If the path id is not specified, 1 is assumed.

A method exists for defining data set relationships independent of the data base structure. This method is used to link a detail data set to a master data in the thread list. This is done by using the LINK option, which specifies an item in the detail data set and is used to perform a calculated access into the specified master data set. This item must match the type and length of the search item in the master data set (which is then the set id following the LINK in the thread list).

All SORT BY and FIND operations work with the current workfile. Executing another WORKFILE IS # deactivates the current workfile and defines a new one. All subsequent SORTs and FINDs then work on the new file. The information in the old workfile is still intact, however, and can be accessed via READ # and PRINT # statements.

Since it may be desirable to return to do additional FINDs and SORTs on the previous workfile, a method is provided for saving and reactivating a workfile. This is done by executing another WORKFILE IS # which does not include the thread list. This will deactivate (but not erase) the current workfile and allow you to activate an old workfile. Do not attempt to reactivate the workfile by re-specifying the thread list, since this loses all information currently in the file by resetting WFLEN to 0.

Expressions are allowed in all WORKFILE IS # parameters. When invoking multiple-line function subprograms, however, these subprograms cannot execute SORT BY, FIND, WORKFILE IS #, IN DATA SET or DBASE ISstatements.

Determining workfile size is a complex function, depending on many factors. A utility program is provided to determine the required file size for a particular application. This program and the formula used are described in Appendix C.

The SORT BY Statement

The SORT BY statement generates pointers to allow accessing data in a specified order.

SORT BY variable name $_1$ [DES] [, ... , variable name $_{10}$ [DES]]

The parameter is:

variable name   A BASIC variable linked via the IN DATA SET statement to an item appearing in one of the data sets in the thread. Substrings are not allowed.

Sorting can occur on up to ten data items[1]; if an order cannot be determined from the first data item, subsequent data items can be specified to determine the order. If no order can be found, the order for those records will be determined by their record pointer value(s) in the data set(s). The specified data is sorted in reverse order by specifying DES. Each data item listed can be sorted in either order.

Data items used for sorting can come from any data set belonging to the thread of the current workfile. When listing the data items in the SORT BY statement, you must place them in order of their significance to the sort, not in their original set order. If an item occurs in two data sets in the thread, the item will be assumed to come from the leftmost set.

[1] Although up to ten items are allowed, the sum of their lengths (in bytes) plus 2 times the thread length must not exceed 256. (Additional overhead is needed for strings with some local-language keyboards; see Appendix C for details.)

After verifying that all parameters are valid, SORT BY copies about 30K bytes (121 sectors) of user memory to the workfile. This allows sorts to use the 30K byte area for work space. The user memory is reloaded upon completion of the sort. Thus, there is a fairly large overhead associated with any sort, regardless of size. If the system is unable to reload the user memory, error 240 is issued and SCRATCH A is performed. Therefore, sorts should not be done in the middle of critical sections where a restart is not possible[2].

Since SORT BY and FIND handle record pointers in the data base, and other users may be modifying the data base, care should be taken when using FIND and SORT BY while the data base is opened in mode 1. Execution of SORT BY requires that all sets in the thread are either read or write locked. Thus, a whole data base lock may be used to achieve the same effect.

After completion of the SORT BY the data sets may be unlocked. However, remember that only pointers are stored in the workfile; if another user (or even the current program) does DBPUTs or DBDELETEs, the pointers may become meaningless due to migrating secondaries in master sets and outright record deletions.

There are a couple of miscellaneous items concerning SORT BY. The first is that executing a SORT BY resets the workfile pointer (as determined by REC) to 1. The second is that if SORT BY is reading the data base via pointers in the workfile (rather than accessing the data base directly) and records in the data base have been deleted since the FIND, SORT or PRINT # that put the pointers there, then any logical workfile record which contains a pointer to a deleted data set record will be deleted. This is true only when SORT BY accesses the set in which the deletion occurred. If there is no sort item needed from that set, SORT BY will not perform the read to determine if a deletion has occurred.

Some example sequences using SORT BY are -

```
SORT BY Order_no$
SORT BY Product_no$,Name$ DES
```

[2] Prior to copying user memory, mode 4 closes are performed on all data bases to insure their integrity. Also, the default mass memory buffer is dumped. Buffers for buffered files and the spooler buffer are not dumped, however.

The FIND Statement

The FIND statement selects a subset of records from the data base thread or the current workfile if the workfile is non-empty.

$$\texttt{FIND} \begin{Bmatrix} \texttt{ALL} \\ \text{condition} \end{Bmatrix}$$

The parameter is:

condition	Any numeric expression used to test variables (or any attribute) for certain conditions. If these conditions are met, the expression has a non-zero (true) result and the record pointers are stored in the workfile. Otherwise, the result is 0 and the record pointers are not stored.

If the workfile has not been used with any previous FIND or SORT BY operation, FIND examines the data base associated with the current workfile. The condition parameter is evaluated to determine whether the group of data entries just read should have their pointers put in the workfile. If the condition is met, the pointers are stored and the next group of entries are processed. Otherwise, the pointers are not stored and processing continued. Note that FIND must actually read each record and trigger the IN DATA SET for each set in the thread to establish the variable values it needs to evaluate the condition expression.

If the workfile already contains pointers (indicated by REC greater than 0), only the data entries specified by the pointers in the workfile are checked by the condition parameter. Pointers to data entries that meet the condition criteria are retained in the workfile; all other pointers are deleted.

Since FIND handles pointers, it is recommended that the data sets in the thread be at least read locked if the data base is opened in mode 1. The sets may also be write locked, or the whole data base may be locked. Unlike SORT BY, however, this is not a requirement.

Specifying FIND ALL is the same as FIND 1=1, and is useful to get all records in unsorted order. If a subsequent FIND or SORT BY is used, however, the FIND ALL is not needed and only wastes time. If a FIND, SORT BY or PRINT # has previously been done, FIND ALL has no effect except to reset the record pointer to record 1.

FIND execution requires the temporary use of from 1032 to 1050 bytes of user memory, depending on thread length.

There are two miscellaneous items concerning FIND. The first is that executing a FIND resets the workfile pointer (as determined by REC) to 1. The second is that if FIND is reading the data base via pointers in the workfile and deletions have occurred in sets involved in the FIND, then FIND will delete the logical workfile records containing pointers to empty data set records.

NOTE

If the condition parameter does not use values from a particular set in the thread (via an IN DATA SET statement), execution time can be improved by deactivating the IN DATA SET statement using the FREE option.

Some example sequences using FIND are:

```
FIND Order_no$>"1000"
FIND (Vendor_no>250) AND (Invoice_no>10000)
FIND ALL
```

The WFLEN Function

The WFLEN function returns the number of logical record pointers contained in the specified workfile.

WFLEN (file number)

The parameter is:

file number   A numeric expression specifying the file number of the workfile.

WFLEN returns a value from 0 through 65534. If a FIND or SORT BY has not been executed on the workfile, 0 is returned. 0 also indicates no entries in the workfile. −1 is returned when the contents of the workfiles are invalid (caused by removing the flexible disc, pressing SHIFT HALT or getting a disc error during a SORT BY or FIND statement). Executing WFLEN on a file other than a workfile causes an error.

The READ # and PRINT # Statements

The READ # and PRINT # statements operate on workfiles in much the same way as they operate on standard BASIC data files. Although their syntax is identical, certain restrictions apply when operating on workfiles.

The first restriction is that only an integeral number of logical records can be read or written. If a partial logical record is read, an error is issued and the record pointer is left at word one of the incompletely read record. If a partial logical record is written, the incompletely written record is not changed; instead, the record pointer is left pointing at the beginning of that record and an error is issued. Strings cannot be read or written on workfiles. Arrays can be written or read by using the array notation (i.e., A(*)) or via MAT PRINT # and MAT READ #.

Note that a pointer value is a value between 1 and the capacity of the set to which it pertains. Since capacities can be as high as 65534, a simple integer variable (or array) may not be able to hold a pointer value. Thus, short or real values should generally be used. The conversion to the appropriate type will be performed automatically.

If a non-integral value is PRINTed on a workfile, it is rounded to an integer. If the rounded value is less than 1 or greater than the set capacity, an error occurs.

The record pointer for READ # and PRINT # can be positioned at any record from 1 through WFLEN +1. Attempting to position past record number WFLEN +1 results in an end-of-file error (which is trappable by ON END #). When printing to records greater than WFLEN, the value of WFLEN is adjusted appropriately. However, actually trying to read values in records beyond WFLEN causes end-of-file error.

PRINTing an END on a workfile resets WFLEN to a value corresponding to the record where END was printed −1. This effectively erases **all** information from the record where END was printed to the end of the workfile.

CHAPTER 3
Program Examples

This chapter shows several programs using SORT/250 operations with the Sales Analysis Data base (SAD). The two programs introduced in Chapter 1 are described here: a program to list products along with their associated orders and a program which also lists the options for each order. Whenever possible, the line numbering for logically-equivalent statements remains the same for each program.

Order List Programs

Each of the order list programs produces a report as shown on page 3-3. This report lists the orders in the data base, broken down by product and organized in sorted order by order number. The products themselves are listed in sorted order. Also, totals are maintained for all orders on each product as well as a total of all orders.

Example program 1 uses a two-set thread (see line 1320). This means that two pointers must be read in line 1480. The R1 pointer refers to a record n the PRODUCT set and the R2 pointer refers to a record in the CUSTOMER set.

Everytime the product changes, the value of R1 also changes. S1 represents the value of R1 at the previous pass through the FOR loop. It is used to detect when it is necessary to print a trailer for the current product (consisting primarily of the total of the orders for the product) and a heading for the new product. Note, however, that printing a trailer at the first pass through the loop is undesirable. A special test for S1=0 is made to stop this from occurring.

Note that the sort performed in line 1360 has Prod_no as its primary sort field. This variable comes from the PRODUCT data set (see line 1190). Because the schema item "PRODUCT-NO" is a search item, however, the value of the variable Product_no$ from the CUSTOMER detail set could just as well have been used.

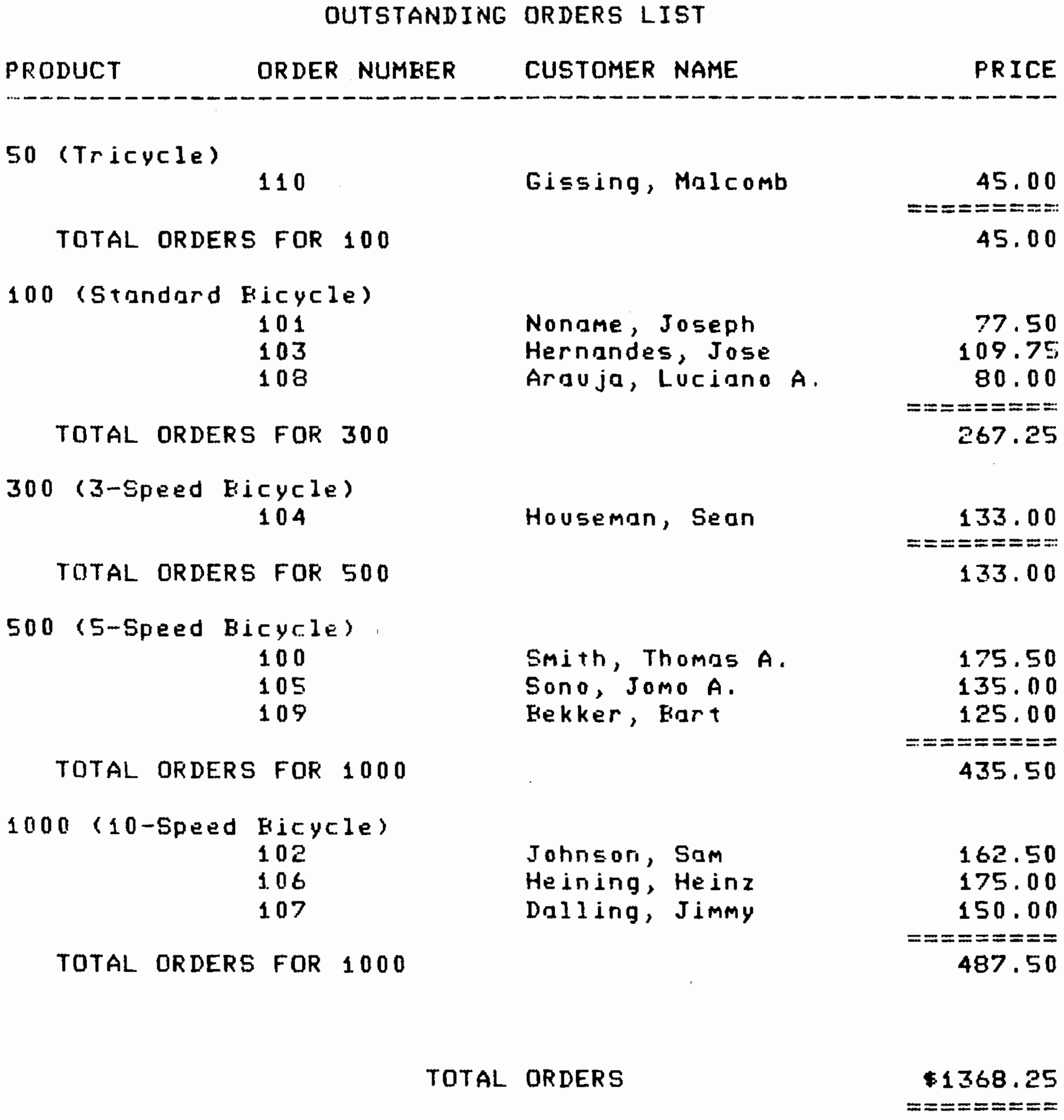
This program shows many poor programming practices which are corrected by example program 2:

- The status array is never tested at any point in the program. The data base may not have been opened; this will ultimately result in error 211 being issued in line 1180.

- As pointed out earlier, the PRODUCT data set need not be involved in the sort. As discussed in Chapter 4, having the PRODUCT data set in the thread greatly reduces efficiency of the SORT BY statement. The description field, however, must be accessed to get the description field for printing. (This is done by a calculated-access DBGET in line 1690 of example program 2.)

- After deleting PRODUCT from the thread there is only one pointer per record in the workfile (see line 1480). This points into the CUSTOMER set, so there is no way to wait for change in recod number to indicate a change in product. Thus, the actual product numbers must be compared. Note that the update of the old product number is accomplished by the IN DATA SET which is triggered when the DBGET in line 1690 is executed. This means that a line analogous to line 1710 in the first example is not needed.

Order List Report



```
                    OUTSTANDING ORDERS LIST

PRODUCT         ORDER NUMBER    CUSTOMER NAME              PRICE
----------------------------------------------------------------


50 (Tricycle)
                110             Gissing, Malcomb           45.00
                                                       =========
   TOTAL ORDERS FOR 100                                    45.00

100 (Standard Bicycle)
                101             Noname, Joseph             77.50
                103             Hernandes, Jose           109.75
                108             Araujo, Luciano A.         80.00
                                                       =========
   TOTAL ORDERS FOR 300                                   267.25

300 (3-Speed Bicycle)
                104             Houseman, Sean            133.00
                                                       =========
   TOTAL ORDERS FOR 500                                   133.00

500 (5-Speed Bicycle)
                100             Smith, Thomas A.          175.50
                105             Sono, Jomo A.             135.00
                109             Bekker, Bart              125.00
                                                       =========
   TOTAL ORDERS FOR 1000                                  435.50

1000 (10-Speed Bicycle)
                102             Johnson, Sam              162.50
                106             Heining, Heinz            175.00
                107             Dalling, Jimmy            150.00
                                                       =========
   TOTAL ORDERS FOR 1000                                  487.50


                          TOTAL ORDERS                  $1368.25
                                                       =========
```

Example Program 1: A Two-set Thread

```
1000 !
1010 !   OUTSTANDING ORDERS REPORT (NOT INCLUDING ALL DETAIL)
1020 !
1030     INTEGER S(9),Prod_no
1040     DIM B$[12],P$[10],Buf$[170]
1050     DIM Desc$[30],Order_no$[30],Name$[30]
1060     DISP "␛K";                                 !  CLEAR SCREEN
1090     B$="  SAD,SALES"
1100     P$="MANAGER"
1110     DBOPEN (B$,P$,1,S(*))                     ! OPEN DATA BASE
1130     DBLOCK (B$,"",1,S(*))                     ! DATA BASE MUST BE LOCKED TO SORT
1150 !
1160 !   SET UP ALL APPROPRIATE RELATIONSHIPS
1170 !
1180     DBASE IS B$
1190     IN DATA SET "PRODUCT" USE Prod_no,Desc$
1200     IN DATA SET "CUSTOMER" USE ALL
1220 !
1230 !   SET UP THE WORKFILE
1240 !
1310     ASSIGN "XYZ" TO #1
1320     WORKFILE IS #1;THREAD IS "PRODUCT","CUSTOMER"
1330 !
1340 !   SORT THE STRUCTURE
1350 !
1360     SORT BY Prod_no,Order_no$
1400 !
1410 !   INITIALIZE VARIABLES & PRINT REPORT HEADER
1420 !
1430 Rep:Total=Master_total=0
1440     S1=0
1450     PRINT TAB(20);"OUTSTANDING ORDERS LIST";LIN(1)
1460     PRINT "PRODUCT";SPA(8);"ORDER NUMBER";SPA(4);"CUSTOMER NAME";SPA(14);
"PRICE";LIN(1);RPT$("-",63);LIN(1)
1461 !
1462 !   PRODUCE THE REPORT
1463 !
1470     FOR Z=1 TO WFLEN(1)
1480       READ #1;R1,R2
1570 !
1580 !     PRINT TRAILER FOR PRODUCT (IF NEEDED)
1590 !
1600 !         (SKIP IF SAME PRODUCT AS BEFORE, OR FIRST TIME THRU LOOP)
1610 !
1620       IF (R1=S1) OR NOT S1 THEN Notot
1630         PRINT USING Tot_image;VAL$(Prod_no),Total
1640         Total=0
1650 !
1660 !     PRINT HEADER FOR PRODUCT (IF NEEDED)
1670 !
1680 Notot:IF R1=S1 THEN Skip1
1690         DBGET (B$,"PRODUCT",4,S(*),"@",Buf$,R1)
1710         S1=R1
1720         PRINT VAL$(Prod_no);" (";TRIM$(Desc$);")"
1810 !
1820 !     PRINT ORDERS
1830 !
1840 Skip1:DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,R2)
1860       PRINT TAB(16);
1870       PRINT USING Itm_image;Order_no$,Name$,Price
1880 Itm_image:IMAGE 16A,22A,2X,5D.DD
1890 !
1900 '     ACCUMULATE TOTALS
1910 !
1920       Total=Total+Price
1940       Master_total=Master_total+Price
1950     NEXT Z
```

(continued)

```
1960 !
1970 !    PRINT FINAL TOTALS
1980 !
2000      PRINT USING Tot_image;VAL$(Prod_no),Total
2010      PRINT USING Mstr_image;Master_total
2040 Tot_image:IMAGE 54X,9("=") / 3X,"TOTAL ORDERS FOR ",10A,24X,6D.DD /
2050 Mstr_image:IMAGE // 25X,"TOTAL ORDERS",14X,"$"8D.DD / 54X,9("=")
2130      END
```

Example Program 2: Using Only One Set Instead of Two

```
1000 !
1010 !    OUTSTANDING ORDERS REPORT (NOT INCLUDING ALL DETAIL)
1020 !
1030      INTEGER S(9),Product_no,Prod_no
1040      DIM B$[12],P$[10],Buf$[170]
1050      DIM Desc$[30],Order_no$[30],Name$[30]
1060      DISP "␛K";                                ! CLEAR SCREEN
1090      B$="  SAD,SALES"
1100      P$="MANAGER"
1110      DBOPEN (B$,P$,1,S(*))                     ! OPEN DATA BASE
1120      IF S(0) THEN Dberr
1130      DBLOCK (B$,"",1,S(*))                     ! DATA BASE MUST BE LOCKED TO SORT
1140      IF S(0) THEN Dberr
1150 !
1160 !    SET UP ALL APPROPRIATE RELATIONSHIPS
1170 !
1180      DBASE IS B$
1190      IN DATA SET "PRODUCT" USE Prod_no,Desc$
1200      IN DATA SET "CUSTOMER" USE ALL
1220 !
1230 !    SET UP THE WORKFILE
1240 !
1310      ASSIGN "XYZ" TO #1
1320      WORKFILE IS #1;THREAD IS "CUSTOMER"
1330 !
1340 !    SORT THE STRUCTURE
1350 !
1360      SORT BY Product_no,Order_no$
1400 !
1410 !    INITIALIZE VARIABLES & PRINT REPORT HEADER
1420 !
1430 Rep:Total=Master_total=0
1440      Prod_no=-1
1450      PRINT TAB(20);"OUTSTANDING ORDERS LIST";LIN(1)
1460      PRINT "PRODUCT";SPA(8);"ORDER NUMBER";SPA(4);"CUSTOMER NAME";SPA(14);
"PRICE";LIN(1);RPT$("-",63);LIN(1)
1461 !
1462 !    PRODUCE THE REPORT
1463 !
1470      FOR Z=1 TO WFLEN(1)
1480        READ #1;R1
1490        DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,R1)
1500        IF S(0) THEN Dberr
1570 !
1580 !      PRINT TRAILER FOR PRODUCT (IF NEEDED)
1590 !
1600 !         (SKIP IF SAME PRODUCT AS BEFORE, OR FIRST TIME THRU LOOP)
1610 !
1620        IF (Prod_no=Product_no) OR (Prod_no<0) THEN Notot
1630          PRINT USING Tot_image;VAL$(Product_no),Total
1640          Total=0
1650 !
1660 !      PRINT HEADER FOR PRODUCT (IF NEEDED)
1670 !
1680 Notot:IF Prod_no=Product_no THEN Skip1
1690          DBGET (B$,"PRODUCT",7,S(*),"@",Buf$,Product_no)
1700          IF S(0) THEN Dberr
1720          PRINT VAL$(Prod_no);" (";TRIM$(Desc$);")"
```

(continued)

```
1810 !
1820 !      PRINT ORDERS
1830 !
1860 Skip1:PRINT TAB(16);
1870       PRINT USING Itm image;Order_no$,Name$,Price
1880 Itm_image:IMAGE 16A,22A,2X,5D.DD
1890 !
1900 !      ACCUMULATE TOTALS
1910 !
1920        Total=Total+Price
1940        Master_total=Master_total+Price
1950      NEXT Z
1960 !
1970 !    PRINT FINAL TOTALS
1980 !
2000      PRINT USING Tot_image;VAL$(Prod_no),Total
2010      PRINT USING Mstr_image;Master_total
2040 Tot image:IMAGE 54X,9("=") / 3X,"TOTAL ORDERS FOR ",10A,24X,6D.DD /
2050 Mstr_image:IMAGE // 25X,"TOTAL ORDERS",14X,"$"8D.DD / 54X,9("=")
2060      STOP
2070 !
2080 !      ERROR TERMINATION ROUTINE
2090 !
2100 Dberr:DISP LIN(2);"STATUS ERROR ";VAL$(S(0));" IN LINE ";S(6)
2170      END
```

Itemized Order List Programs

The remaining three programs are all extensions to the previous programs, in that the report is essentially the same, but each order has its options listed along with it. In example programs 3 and 4 the options are listed in sorted order. A report that could be printed by these programs is shown on pages 3-8/9. Example program 5 lists the options in the order they occur along the chain in the OPTION detail. The report produced by this program is shown on pages 3-13/14.

Note that there is a blank option following the customer name. There is actually an entry with a blank option number field in ORDER for each order placed. This record contains the price of the product, and the all-blank field is used to force this entry to occur before any of the other options when sorted. It is also put into the detail before any of the options to guarantee that it will be the first in the chain.

The blank entry also serves another function. If it were not included, then any order sold with no options would have no record in the OPTION set. This would generate an incomplete hierarchy for such orders, so they would not occur in the workfile generated by programs 3 and 4, though program 5 could be modified to handle such orders.

Example program 3 uses a four-set thread (see line 1320). The construction of this thread is discusses in Chapter 1. Note, however, that although four pointers must be read from the workfile (see line 1480), the third pointer, R3, is never used. This third pointer is just the place holder to skip over the information in the automatic set, ORDER. Again, the change in record number pertaining to the PRODUCT set is used to trigger the headers and trailers for new products (via variables R1 and S1). A similar technique is used to detect the change in order number (via variables R2 and S2).

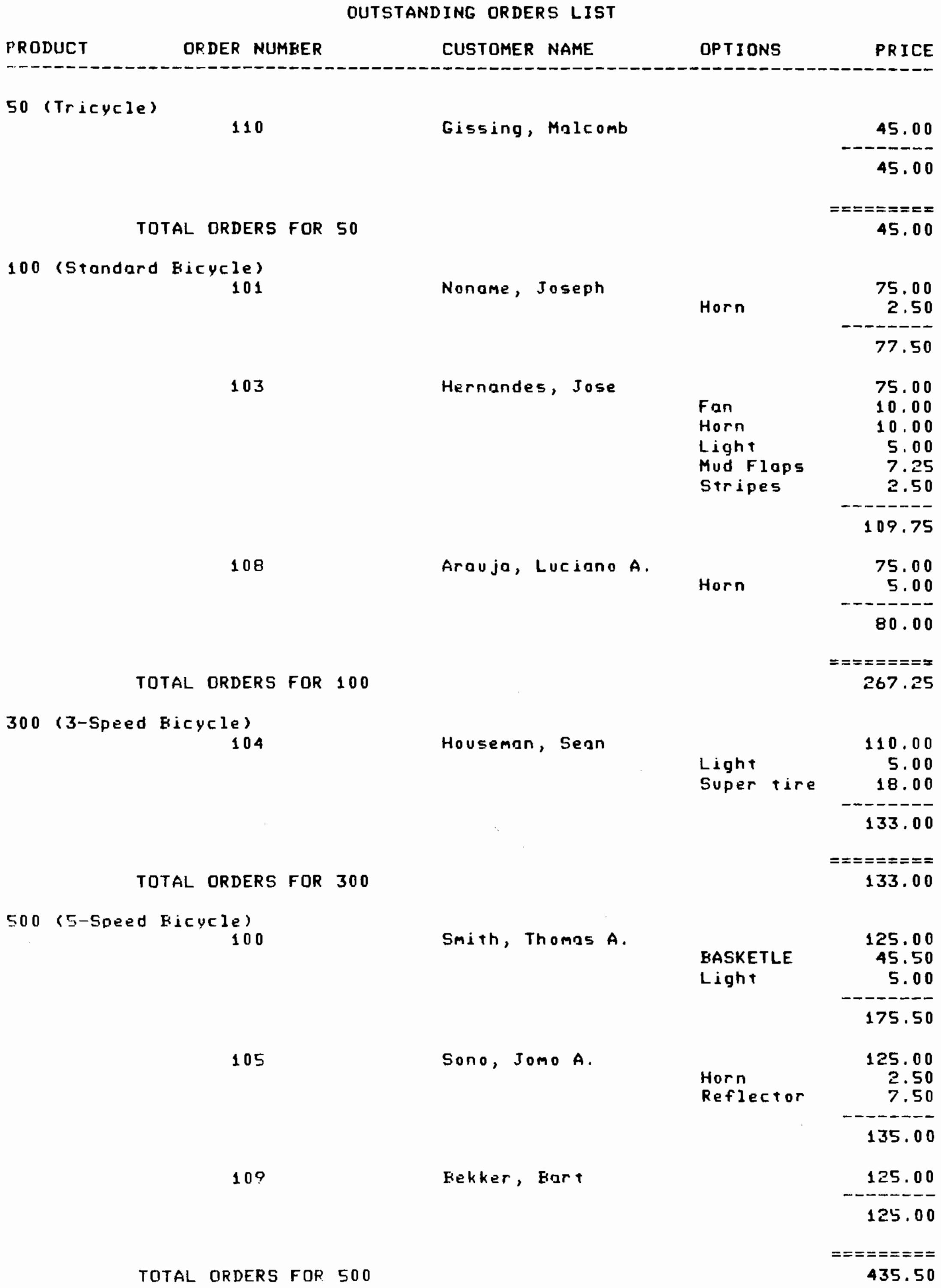
Example program 3 is another case of bad programming. Example program 4 cleans up these problems. It adds status checks for data base calls, error trapping (see line 1070) and HALT key trapping (see line 1080). Also, all the previous examples have assumed that the data file "XYZ" existed for use as a workfile. Example program 4 now checks to see if the workfile exists and creates it if it does not. It stops if the file is protected or is of the wrong type.

For reasons detailed in Chapter 4, long threads are undesirable and should be avoided when possible. As in example program 2, the PRODUCT set can be eliminated from the thread by use of a calculated-access DBGET. This reduces the thread length to three. Also, if it is not particularly important to have the options listed in sorted order, a DBFIND on the OPTION set using the order number from the CUSTOMER set may be done. This allows chained mode DBGETs to be used to get the options. Listing will thus be in the chain order (the order the options appeared in on the original order). This reduces the thread length to only one set, the CUSTOMER set. Program example 5 shows how this could be done.

In example program 5, as in example 2, the actual product number is used to determine when product headers and trailers are required. However, since each record in the workfile corresponds to a new order, no special logic is needed to detect change in order number; the header and trailer each occur every time through the loop. A special imbedded FOR loop is added, however, to print out the options (see lines 1835 through 1945).

Example program 5 does not lock the data base. It only locks the sets in the thread and the sets used in the report. This allows multiple users to simultaneously run copies of the program from different consoles. If a data base lock had been used, one program would run while others would wait at DBLOCK for access to the data base.

Itemized Options List Report (sorted order)



```
                              OUTSTANDING ORDERS LIST

PRODUCT          ORDER NUMBER          CUSTOMER NAME          OPTIONS          PRICE
-------------------------------------------------------------------------------------

50 (Tricycle)
                     110               Gissing, Malcomb                        45.00
                                                                             --------
                                                                               45.00

                                                                            =========
            TOTAL ORDERS FOR 50                                                45.00

100 (Standard Bicycle)
                     101               Noname, Joseph                          75.00
                                                              Horn              2.50
                                                                             --------
                                                                               77.50

                     103               Hernandes, Jose                         75.00
                                                              Fan              10.00
                                                              Horn             10.00
                                                              Light             5.00
                                                              Mud Flaps         7.25
                                                              Stripes           2.50
                                                                             --------
                                                                              109.75

                     108               Araujo, Luciano A.                      75.00
                                                              Horn              5.00
                                                                             --------
                                                                               80.00

                                                                            =========
            TOTAL ORDERS FOR 100                                              267.25

300 (3-Speed Bicycle)
                     104               Houseman, Sean                         110.00
                                                              Light             5.00
                                                              Super tire       18.00
                                                                             --------
                                                                              133.00

                                                                            =========
            TOTAL ORDERS FOR 300                                              133.00

500 (5-Speed Bicycle)
                     100               Smith, Thomas A.                       125.00
                                                              BASKETLE         45.50
                                                              Light             5.00
                                                                             --------
                                                                              175.50

                     105               Sono, Jomo A.                          125.00
                                                              Horn              2.50
                                                              Reflector         7.50
                                                                             --------
                                                                              135.00

                     109               Bekker, Bart                           125.00
                                                                             --------
                                                                              125.00

                                                                            =========
            TOTAL ORDERS FOR 500                                              435.50
```

(continued)

```
1000 (10-Speed Bicycle)
                   102              Johnson, Sam                        150.00
                                                        Chrome           12.50
                                                                       --------
                                                                        162.50

                   106              Heining, Heinz                      150.00
                                                        Basket           15.00
                                                        Light            10.00
                                                                       --------
                                                                        175.00

                   107              Dalling, Jimmy                      150.00
                                                                       --------
                                                                        150.00

                                                                      =========
           TOTAL ORDERS FOR 1000                                        487.50


                              TOTAL ORDERS                            $1368.25
                                                                      =========
```

Example Program 3: A Four-set Thread

```
1000 !
1010 !    OUTSTANDING ORDERS REPORT (INCLUDING ALL DETAIL)
1020 !
1030      INTEGER S(9),Prod_no
1040      DIM B$[12],P$[10],Buf$[170]
1050      DIM Desc$[30],Order_no$[30],Name$[30],Option_desc$[10]
1060      DISP "␛E";                              ! CLEAR SCREEN
1090      B$="  SAD,SALES"
1100      P$="MANAGER"
1110      DBOPEN (B$,P$,1,S(*))                   ! OPEN DATA BASE
1130      DBLOCK (B$,"",1,S(*))                   ! DATA BASE MUST BE LOCKED TO SORT
1150 !
1160 !    SET UP ALL APPROPRIATE RELATIONSHIPS
1170 !
1180      DBASE IS B$
1190      IN DATA SET "PRODUCT" USE Prod_no,Desc$
1200      IN DATA SET "CUSTOMER" USE ALL
1210      IN DATA SET "OPTION" USE SKP 1,Option_desc$,P0
1220 !
1230 !    SET UP THE WORKFILE
1240 !
1310      ASSIGN "XYZ" TO #1
1320      WORKFILE IS #1;THREAD IS "PRODUCT","CUSTOMER","ORDER","OPTION"
1330 !
1340 !    SORT THE STRUCTURE
1350 !
1360      SORT BY Prod_no,Order_no$,Option_desc$
1400 !
1410 !    INITIALIZE VARIABLES & PRINT REPORT HEADER
1420 !
1430 Rep:Sub_total=Total=Master_total=0
1440      S1=S2=0
1450      PRINT TAB(30);"OUTSTANDING ORDERS LIST";LIN(1)
1460      PRINT "PRODUCT";SPA(8);"ORDER NUMBER";SPA(10);"CUSTOMER NAME";SPA(9);
"OPTIONS";SPA(8);"PRICE";LIN(1);RPT$("-",79);LIN(1)
```

(continued)

```
1461 !
1462 !   PRODUCE THE REPORT
1463 !
1470     FOR Z=1 TO WFLEN(1)
1480       READ #1;R1,R2,R3,R4
1490 !
1500 !     PRINT TRAILER FOR ORDER (IF NEEDED)
1510 !
1520 !         (SKIP IF SAME ORDER AS BEFORE, OR FIRST TIME THRU LOOP)
1530 !
1540       IF (R2=S2) OR NOT S2 THEN Nosub
1550         PRINT USING Sub_image;Sub_total
1560         Sub_total=0
1570 !
1580 !     PRINT TRAILER FOR PRODUCT (IF NEEDED)
1590 !
1600 !         (SKIP IF SAME PRODUCT AS BEFORE, OR FIRST TIME THRU LOOP)
1610 !
1620 Nosub:IF (R1=S1) OR NOT S1 THEN Notot
1630         PRINT USING Tot_image;VAL$(Prod_no),Total
1640         Total=0
1650 !
1660 !     PRINT HEADER FOR PRODUCT (IF NEEDED)
1670 !
1680 Notot:IF R1=S1 THEN Skip1
1690         DBGET (B$,"PRODUCT",4,S(*),"@",Buf$,R1)
1710         S1=R1
1720         PRINT VAL$(Prod_no);" (";TRIM$(Desc$);")"
1730 !
1740 !     PRINTER HEADER FOR ORDER (IF NEEDED)
1750 !
1760 Skip1:IF R2=S2 THEN Skip2
1770         DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,R2)
1790         PRINT TAB(20);Order_no$;TAB(38);Name$[1,21];
1800         S2=R2
1810 !
1820 !     PRINT OPTIONS
1830 !
1840 Skip2:DBGET (B$,"OPTION",4,S(*),"@",Buf$,R4)
1860       PRINT TAB(60);
1870       PRINT USING Itm_image;Option_desc$,P0
1880 Itm_image:IMAGE 10A,2X,5D.DD
1890 !
1900 !     ACCUMULATE TOTALS
1910 !
1920       Total=Total+P0
1930       Sub_total=Sub_total+P0
1940       Master_total=Master_total+P0
1950     NEXT Z
1960 !
1970 !   PRINT FINAL TOTALS
1980 !
1990     PRINT USING Sub_image;Sub_total
2000     PRINT USING Tot_image;VAL$(Prod_no),Total
2010     PRINT USING Mstr_image;Master_total
2030 Sub_image:IMAGE 71X,8("-") / 71X,5D.DD /
2040 Tot_image:IMAGE 70X,9("=") / 11X,"TOTAL ORDERS FOR ",10A,32 X,6D.DD /
2050 Mstr_image:IMAGE // 31X,"TOTAL ORDERS",24X,"$"8D.DD / 70X,9("=")
2160     END
```

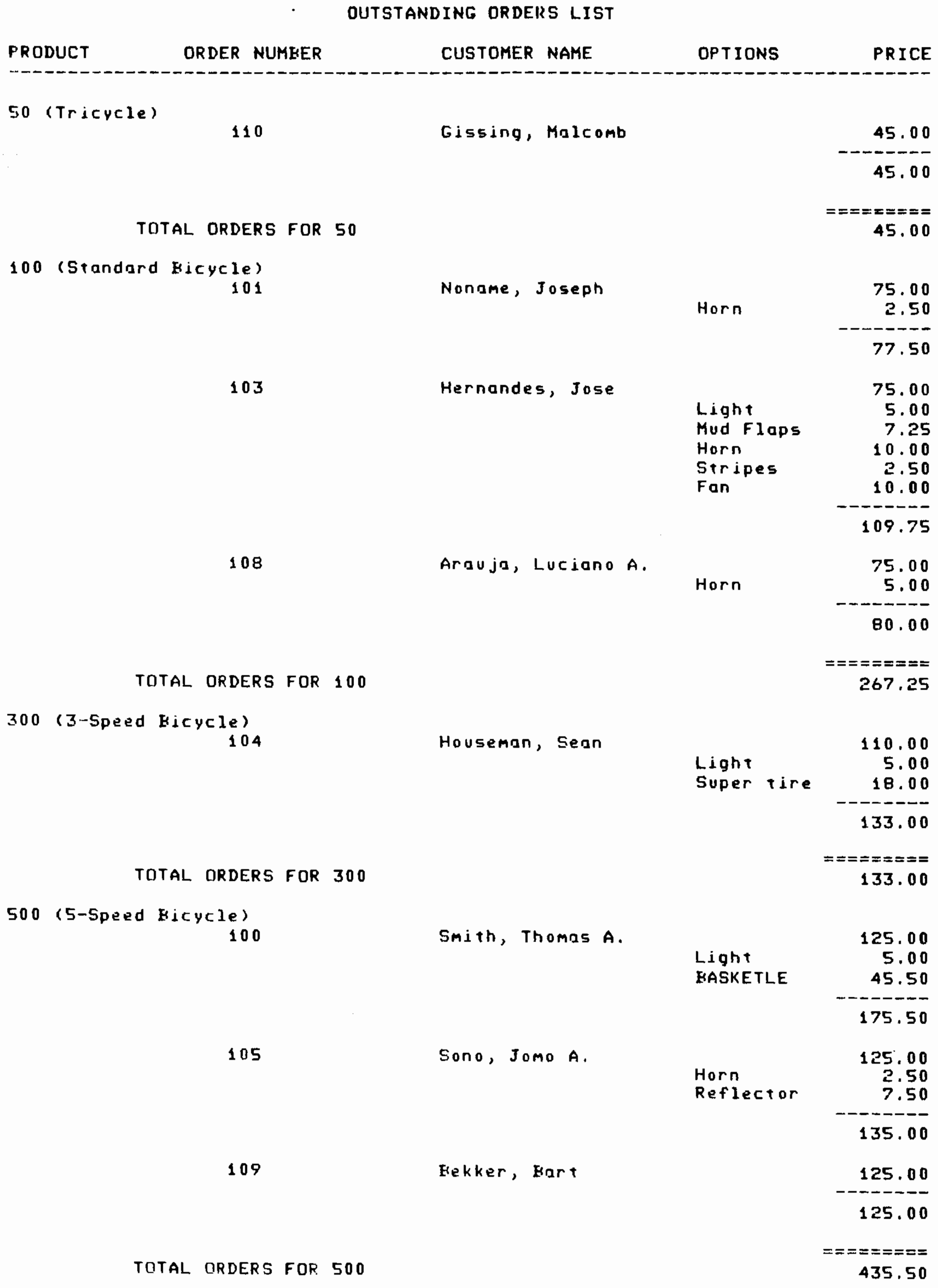
Example Program 4: Using Only One Set Instead of Four

```
1000 !
1010 !   OUTSTANDING ORDERS REPORT (INCLUDING ALL DETAIL)
1020 !
1030     INTEGER S(9),Prod_no
1040     DIM B$[12],P$[10],Buf$[170]
1050     DIM Desc$[30],Order_no$[30],Name$[30],Option_desc$[10]
1060     DISP "␛E";                             ! CLEAR SCREEN
1070     ON ERROR GOTO Error                    ! SET UP ERROR AND HALT TRAPS
1080     ON HALT GOTO Halt
1090     B$="  SAD,SALES"
1100     P$="MANAGER"
1110     DBOPEN (B$,P$,1,S(*))                  ! OPEN DATA BASE
1120     IF S(0) THEN Dberr
1130     DBLOCK (B$,"",1,S(*))                  ! DATA BASE MUST BE LOCKED TO SORT
1140     IF S(0) THEN Dberr
1150 !
1160 !   SET UP ALL APPROPRIATE RELATIONSHIPS
1170 !
1180     DBASE IS B$
1190     IN DATA SET "PRODUCT" USE Prod_no,Desc$
1200     IN DATA SET "CUSTOMER" USE ALL
1210     IN DATA SET "OPTION" USE SKP 1,Option_desc$,P0
1220 !
1230 !   SET UP THE WORKFILE
1240 !
1250     ASSIGN "XYZ" TO #1,Z
1260     IF Z<2 THEN Ok
1270     DISP "CAN'T ASSIGN THE WORKFILE!"
1280     STOP
1290 Ok: IF NOT Z THEN Aok                      ! CREATE WORK FILE IF NECESSARY
1300     FCREATE "XYZ",130
1310     ASSIGN "XYZ" TO #1
1320 Aok:WORKFILE IS #1;THREAD IS "PRODUCT","CUSTOMER","ORDER","OPTION"
1330 !
1340 !   SORT THE STRUCTURE
1350 !
1360     SORT BY Prod_no,Order_no$,Option_desc$
1370     IF WFLEN(1) THEN Rep
1380     DISP "THERE ARE NO ENTRIES IN THE STRUCTURE TO REPORT ON."
1390     STOP
1400 !
1410 !   INITIALIZE VARIABLES & PRINT REPORT HEADER
1420 !
1430 Rep:Sub_total=Total=Master_total=0
1440     S1=S2=0
1450     PRINT TAB(30);"OUTSTANDING ORDERS LIST";LIN(1)
1460     PRINT "PRODUCT";SPA(8);"ORDER NUMBER";SPA(10);"CUSTOMER NAME";SPA(9);
"OPTIONS";SPA(8);"PRICE";LIN(1);RPT$("-",79);LIN(1)
1461 !
1462 !   PRODUCE THE REPORT
1463 !
1470     FOR Z=1 TO WFLEN(1)
1480       READ #1;R1,R2,R3,R4
1490 !
1500 !     PRINT TRAILER FOR ORDER (IF NEEDED)
1510 !
1520 !        (SKIP IF SAME ORDER AS BEFORE, OR FIRST TIME THRU LOOP)
1530 !
1540       IF (R2=S2) OR NOT S2 THEN Nosub
1550         PRINT USING Sub_image;Sub_total
1560         Sub_total=0
1570 !
1580 !     PRINT TRAILER FOR PRODUCT (IF NEEDED)
1590 !
1600 !        (SKIP IF SAME PRODUCT AS BEFORE, OR FIRST TIME THRU LOOP)
1610 !
1620 Nosub:IF (R1=S1) OR NOT S1 THEN Notot
1630         PRINT USING Tot_image;VAL$(Prod_no),Total
1640         Total=0
```

(continued)

```
1650 !
1660 !     PRINT HEADER FOR PRODUCT (IF NEEDED)
1670 !
1680 Notot:IF R1=S1 THEN Skip1
1690         DBGET (B$,"PRODUCT",4,S(*),"@",Buf$,R1)
1700         IF S(0) THEN Dberr
1710         S1=R1
1720         PRINT VAL$(Prod_no);" (";TRIM$(Desc$);")"
1730 !
1740 !     PRINTER HEADER FOR ORDER (IF NEEDED)
1750 !
1760 Skip1:IF R2=S2 THEN Skip2
1770         DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,R2)
1780         IF S(0) THEN Dberr
1790         PRINT TAB(20);Order_desc$;TAB(38);Name$[1,21];
1800         S2=R2
1810 !
1820 !     PRINT OPTIONS
1830 !
1840 Skip2:DBGET (B$,"OPTION",4,S(*),"@",Buf$,R4)
1850       IF S(0) THEN Dberr
1860       PRINT TAB(60);
1870       PRINT USING Itm_image;Option_no$,P0
1880 Itm_image:IMAGE 10A,2X,5D.DD
1890 !
1900 !     ACCUMULATE TOTALS
1910 !
1920       Total=Total+P0
1930       Sub_total=Sub_total+P0
1940       Master_total=Master_total+P0
1950     NEXT Z
1960 !
1970 !   PRINT FINAL TOTALS
1980 !
1990     PRINT USING Sub_image;Sub_total
2000     PRINT USING Tot_image;VAL$(Prod_no),Total
2010     PRINT USING Mstr_image;Master_total
2020     DISP "REPORT COMPLETE."
2030 Sub_image:IMAGE 71X,8("-") / 71X,5D.DD /
2040 Tot_image:IMAGE 70X,9("=") / 11X,"TOTAL ORDERS FOR ",10A,32 X,6D.DD /
2050 Mstr_image:IMAGE // 31X,"TOTAL ORDERS",24X,"$"8D.DD / 70X,9("=")
2060     STOP
2070 !
2080 !     ERROR AND HALT TERMINATION ROUTINES
2090 !
2100 Dberr:DISP LIN(2);"STATUS ERROR ";VAL$(S(0));" IN LINE";S(6)
2110     STOP
2120 Error:DISP LIN(2);"UNEXPECTED ";ERRM$
2130     STOP
2140 Halt:PRINT LIN(2)
2150     DISP LIN(2);"PROGRAM TERMINATED."
2160     END
```

Itemized Options List Report (unsorted order)



```
                             OUTSTANDING ORDERS LIST

PRODUCT        ORDER NUMBER          CUSTOMER NAME        OPTIONS        PRICE
------------------------------------------------------------------------------

50 (Tricycle)
                   110                 Gissing, Malcomb                    45.00
                                                                      --------
                                                                         45.00

                                                                     =========
           TOTAL ORDERS FOR 50                                           45.00

100 (Standard Bicycle)
                   101                 Noname, Joseph                      75.00
                                                          Horn            2.50
                                                                      --------
                                                                         77.50

                   103                 Hernandes, Jose                     75.00
                                                          Light           5.00
                                                          Mud Flaps       7.25
                                                          Horn           10.00
                                                          Stripes         2.50
                                                          Fan            10.00
                                                                      --------
                                                                        109.75

                   108                 Arauja, Luciano A.                  75.00
                                                          Horn            5.00
                                                                      --------
                                                                         80.00

                                                                     =========
           TOTAL ORDERS FOR 100                                         267.25

300 (3-Speed Bicycle)
                   104                 Houseman, Sean                     110.00
                                                          Light           5.00
                                                          Super tire     18.00
                                                                      --------
                                                                        133.00

                                                                     =========
           TOTAL ORDERS FOR 300                                         133.00

500 (5-Speed Bicycle)
                   100                 Smith, Thomas A.                   125.00
                                                          Light           5.00
                                                          BASKETLE       45.50
                                                                      --------
                                                                        175.50

                   105                 Sono, Jomo A.                      125.00
                                                          Horn            2.50
                                                          Reflector       7.50
                                                                      --------
                                                                        135.00

                   109                 Bekker, Bart                       125.00
                                                                      --------
                                                                        125.00

                                                                     =========
           TOTAL ORDERS FOR 500                                         435.50
```

(continued)

```
1000 (10-Speed Bicycle)
                     102              Johnson, Sam                             150.00
                                                               Chrome           12.50
                                                                             --------
                                                                               162.50

                     106              Heining, Heinz                           150.00
                                                               Light            10.00
                                                               Basket           15.00
                                                                             --------
                                                                               175.00

                     107              Dalling, Jimmy                           150.00
                                                                             --------
                                                                               150.00

                                                                            =========
            TOTAL ORDERS FOR 1000                                              487.50


                                  TOTAL ORDERS                               $1368.25
                                                                            =========
```

Example Program 5: Listing Options in Unsorted Order

```
1000 !
1010 !    OUTSTANDING ORDERS REPORT (INCLUDING ALL DETAIL)
1020 !
1030      INTEGER S(9),Product_no,Prod_no
1040      DIM B$[12],P$[10],Buf$[170]
1050      DIM Desc$[30],Order_no$[30],Name$[30],Option_desc$[10]
1060      DISP "␛E";                            ! CLEAR SCREEN
1070      ON ERROR GOTO Error                   ! SET UP ERROR AND HALT TRAPS
1080      ON HALT GOTO Halt
1090      B$="  SAD,SALES"
1100      P$="MANAGER"
1110      DBOPEN (B$,P$,1,S(*))                 ! OPEN DATA BASE
1120      IF S(0) THEN Dberr
1125      PREDICATE Buf$ FROM "PRODUCT","CUSTOMER","OPTION"
1130      DBLOCK (B$,Buf$,13,S(*))
1140      IF S(0) THEN Dberr
1150 !
1160 !    SET UP ALL APPROPRIATE RELATIONSHIPS
1170 !
1180      DBASE IS B$
1190      IN DATA SET "PRODUCT" USE Prod_no,Desc$
1200      IN DATA SET "CUSTOMER" USE ALL
1210      IN DATA SET "OPTION" USE SKP 1,Option_desc$,P0
1220 !
1230 !    SET UP THE WORKFILE
1240 !
1250      ASSIGN "XYZ" TO #1,Z
1260      IF Z<2 THEN Ok
1270      DISP "CAN'T ASSIGN THE WORKFILE!"
1280      STOP
1290 Ok:  IF NOT Z THEN Aok                     ! CREATE WORK FILE IF NECESSARY
1300      FCREATE "XYZ",130
1310      ASSIGN "XYZ" TO #1
1320 Aok:WORKFILE IS #1;THREAD IS "CUSTOMER"
1330 !
1340 !    SORT THE STRUCTURE
1350 !
1360      SORT BY Product_no,Order_no$
1370      IF WFLEN(1) THEN Rep
1380      DISP "THERE ARE NO ENTRIES IN THE STRUCTURE TO REPORT ON."
1390      STOP
```

(continued)

```
1400 !
1410 !   INITIALIZE VARIABLES & PRINT REPORT HEADER
1420 !
1430 Rep:Total=Master_total=0
1440     Prod_no=-1
1450     PRINT TAB(30);"OUTSTANDING ORDERS LIST";LIN(1)
1460     PRINT "PRODUCT";SPA(8);"ORDER NUMBER";SPA(10);"CUSTOMER NAME";SPA(9);
"OPTIONS";SPA(8);"PRICE";LIN(1);RPT$("-",79);LIN(1)
1461 !
1462 !   PRODUCE THE REPORT
1463 !
1470     FOR Z=1 TO WFLEN(1)
1480       READ #1;R1
1490       DBGET (B$,"CUSTOMER",4,S(*),"@",Buf$,R1)
1500       IF S(0) THEN Dberr
1520 !          (SKIP IF SAME ORDER AS BEFORE, OR FIRST TIME THRU LOOP)
1530 !
1570 !
1580 !     PRINT TRAILER FOR PRODUCT (IF NEEDED)
1590 !
1600 !        (SKIP IF SAME PRODUCT AS BEFORE, OR FIRST TIME THRU LOOP)
1610 !
1620 Nosub:IF (Prod_no=Product_no) OR (Prod_no(0) THEN Notot
1630         PRINT USING Tot_image;VAL$(Prod_no),Total
1640         Total=0
1650 !
1660 !     PRINT HEADER FOR PRODUCT (IF NEEDED)
1670 !
1680 Notot:IF Prod_no=Product_no THEN Skip1
1690         DBGET (B$,"PRODUCT",7,S(*),"@",Buf$,Product_no)
1700         IF S(0) THEN Dberr
1720         PRINT VAL$(Prod_no);" (";TRIM$(Desc$);")"
1730 !
1740 !     PRINT HEADER FOR ORDER
1750 !
1790 Skip1:PRINT TAB(20);Order_no$;TAB(38);Name$[1,21];
1810 !
1820 !     PRINT OPTIONS
1830 !
1835       DBFIND (B$,"OPTION",1,S(*),"ORDER-NO",Order_no$)
1836       IF S(0) THEN Dberr
1840       FOR C=1 TO S(5)
1845         DBGET (B$,"OPTION",5,S(*),"@",Buf$,0)
1850         IF S(0) THEN Dberr
1860         PRINT TAB(60);
1870         PRINT USING Itm_image;Option_desc$,P0
1880 Itm_image:IMAGE 10A,2X,5D.DD
1890 !
1900 !     ACCUMULATE TOTALS
1910 !
1920         Total=Total+P0
1930         Sub_total=Sub_total+P0
1940         Master_total=Master_total+P0
1945       NEXT C
1946       PRINT USING Sub_image;Sub_total
1947       Sub_total=0
1950     NEXT Z
1960 !
1970 !   PRINT FINAL TOTALS
1980 !
2000     PRINT USING Tot_image;VAL$(Prod_no),Total
2010     PRINT USING Mstr_image;Master_total
2020     DISP "REPORT COMPLETE."
2030 Sub_image:IMAGE 71X,8("-") / 71X,5D.DD /
2040 Tot_image:IMAGE 70X,9("=") / 11X,"TOTAL ORDERS FOR ",10A,32 X,6D.DD /
2050 Mstr_image:IMAGE // 31X,"TOTAL ORDERS",24X,"$"8D.DD / 70X,9("=")
2060     STOP
2070 !
2080 !     ERROR AND HALT TERMINATION ROUTINES
2090 !
2100 Dberr:DISP LIN(2);"STATUS ERROR ";VAL$(S(0));" IN LINE";S(6)
2110     STOP
2120 Error:DISP LIN(2);"UNEXPECTED ";ERRM$
2130     STOP
2140 Halt:PRINT LIN(2)
2150     DISP LIN(2);"PROGRAM TERMINATED."
2160     END
```

CHAPTER 4

Programming Considerations

Introduction

A great deal can be done toward speeding up SORT/250 operations by following certain programming guidelines and by proper layout of data sets on the media. This chapter presents factors which should be considered in program and data base design to optimize sorting speed. Use of some factors will always result in optimum sort speed. Use of other factors may increase or decrease speed, depending on how they are implemented; trial and error will determine the optimum combination for a given application.

Data Set/File Layout

Perhaps the most difficult parameters to optimize are those associated with file placement on the disc. There are basically four parameters which effect disc I/O time:

Head settling time - the time needed to wait for oscillations in the floppy head to damp out after dropping onto the media.

Seek time - the time necessary for the head to move from the track it is currently on to the track where the required sector(s) reside.

Latency time - the time necessary to wait for the desired sector to be located underneath the read head (worst case is one revolution of the disc).

Interleave factor - a measure of the number of physical sectors between two logically-sequential sectors.

In general, control of the latency time is extremely difficult and not a factor worth considering. Interleave factor is determined at floppy initialization; the default interleave factor of four is optimized for the HP250 System. The interleave factor for hard discs is not user controllable. Thus, only head settling time and seek time are controllable.

Head settling time is only an issue on flexible discs, and then only when all data sets in the thread and the workfile are not on the same volume. If more than one volume is used in the sort, because of a multi-set thread, the data sets should be split up so that the sets involved in the SORT or FIND alternate between the available volumes. For the case where no FINDs or SORTs have been done on the workfile (REC=0), all sets are involved in the operation and should be alternated between the volumes. If the workfile has pointers in it already, only the data sets from which a sort item originates (in the case of SORT BY) and only those data sets which have an IN DATA SET active on them (in the case of FIND) are considered to be actively involved in the SORT or FIND. Only those sets involved need to be alternated.

Clearly, it may not be possible to satisfy the above criteria when more than just a single FIND or SORT is done. In this case, experimental placement of the data sets may be done, although it may not be worthwhile when relatively few head lifts occur.

The other factor which slows down execution speed is head movement. There are two simple rules to follow to minimize head movement. First, insure that the DBCREATE is issued on an essentially "repacked" disc or that all but one of the unused areas on the disc are smaller than the smallest data set. This ensures that the data set files will be created one after the another on the media (regardless of physically where their names appear in the directory). Thus, no intervening files are present to lengthen the seek.

The second rule is somewhat more complicated. It involves knowing what kinds of FINDs and SORTs will be done. Basically, the sets in the schema should be created (via DBCREATE), so that sets occurring on the same volume which will be involved in the same thread are as close as possible on the disc. This reduces the number of intervening sets and, thus, the length of the seek.

One comment relative to workfile placement: the workfile should be kept as small as possible while still ensuring that it has sufficient size (see Appendix C for details). Certain information in the workfile (mostly of a temporary nature) is stored near the end of the file during a SORT BY. Pointers are stored near the beginning of the file. If the workfile is too large, extra tracks will have to be skipped to get from the pointers to this temporary information. Also, note that if the workfile is purged and re-created for each use, it may be placed at different parts of the disc, thus producing varying execution times due to random head movement.

Software Optimizations

The most significant gains in terms of speed improvement can be made by following some simple rules in designing programs using SORT/250 operations. There are essentially three classes of rules which will be covered:

- Generally true rules.
- Rules which are to be used if no FINDs, SORTs or PRINT #s have been done on the workfile (REC=0).
- Rules which are to be used if pointers have been put in the workfile (REC≠0).

General Rules

The most important rule is to keep thread length minimal. In fact, if thread length can be kept to 1, the SORT/250 System will use a large number of special optimizations that cannot be otherwise used. Also, if either the first or the last set in the thread is a master and the only item that will be ever used out of it for FINDing or SORTing is the search item, it can be eliminated. This is possible since that item also exists in the associated detail thus enabling a calculated access DBGET to be used to get the additional information out of the master.

The second rule deals with execution of SORT BY. The function for execution time of SORT BY is only piece-wise linear. For small sorts[1] the time is generally dependent on the time needed to read the entries from the data sets in the thread. If a merge is required, however, time increases dramatically. There is no precise formula for describing sort times; however, it can be assumed to be the same in terms of shape as the workfile size graph (see Appendix C). Discontinuities in the execution time function will occur at the same places as they do in the workfile size function.

The last rule is to **turn off** all possible IN DATA SETs (via the FREE option) before doing a FIND. If a particular IN DATA SET is active for some data set in the thread and no values from that set are needed to evaluate the selection expression, FREEing that IN DATA SET stops FIND from reading information from that set.

[1] A small sort is defined as one which can be performed entirely in memory (i.e., no merges will be required [this can be detected via use of the SIZE GRAPH function in the WORK program described in Appendix C]).

Rules When REC = 0

If an item occurs in more than one set in the thread (generally because it is a search item), it should be selected to come from the set closest to the start of the thread.For FIND it is very important to notice that the value of the conditional expression cannot be evaluated until all appropriate sets (ones with IN DATA SETs active) have been read. Thus, if the needed set can be restricted to those near the head of the thread, the expression can be evaluated sooner.[2]

When there are no pointers in the workfile, SORT BY and FIND statements must do a serial scan of the first set in the thread. This means the execution time of a SORT BY or FIND is not strictly dependent of the number of entries, but where the last non-empty record in the set occurs. In the case of master data sets where, despite a small number of entries, the last record may be very near the end of the set, the increase in execution time is particularly pronounced.

Rules When REC ≠ 0

Here, again, it is a good idea to deactivate all unused IN DATA SET relations pertaining to sets in the thread. In the case of SORT BY, the fewer sets involved the better. Remember that if one of the sort items is a search item it may be possible to select it from one of several sets. Select it from the set which allows you to deactivate the most IN DATA SETs.

In the case of FIND the same things as mentioned for SORT BY also apply. However, breaking up a complex FIND separated by ANDs into several FINDs may increase speed if (and only if) some of the clauses separated by the ANDs do not involve the same sets or involve fewer sets than the other clauses. If this is the case, the clauses which have the fewest sets involved and the lowest probability of being true should be executed first. Remember, again, that the only way FIND knows which sets are involved is by which IN DATA SETs are active.

Clearly, most of these rules assume the programmer has a good understanding of the form the data will take (in terms of probable events). When in doubt, perform tests.

	REC = 0 (no previous FIND, SORT BY or PRINT #)	**REC ≠ 0** (previous FIND, SORT BY or PRINT #)
FIND	Keep thread length short. Make sure the last set with an IN DATA SET active on it is as close to the start of the thread as possible.	Make sure IN DATA SETS are active on only those sets from which information must be retrieved.
SORT BY	Keep thread length short.	Make sure sort keys come from as few sets as possible.

Always:

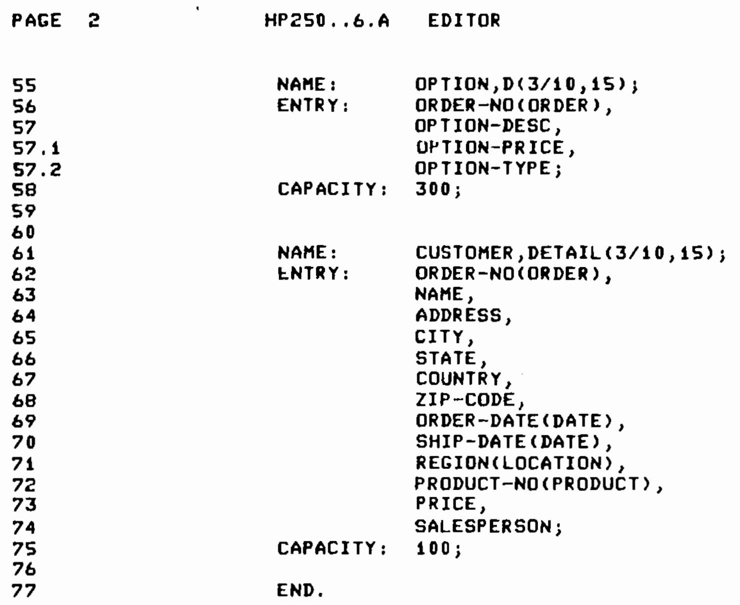
1) Minimize thread length.
2) Minimize complexity of the FIND selection expression.
3) Minimize total sort key.

[2] If the FIND condition is a series of conditions separated by ANDs, it may be beneficial to break them up into separate FINDs. In general, if some of the clauses pertain only to the first set in the thread and they will select significantly less than all the data available, then it is best to construct two FIND statements (the first one pertaining only to the set at the head of the thread). Remember when doing this to deactivate and reactivate the IN DATA SET relations (via FREE) to maximize effect.

APPENDIX A

Schema Listing for the SAD Data Base

```
PAGE  1              HP250.A   EDITOR

     $CONTROL           LIST,TABLE,ROOT
     $TITLE  "Sales Analysis Data Base"

     BEGIN DATA BASE    SAD;  <<CUSTOMER SALES ANALYSIS DATA BASE>>

     PASSWORDS:
                        10           SALESMAN;
                        15           MANAGER;
                         3           SECRTARY; <<WILL HAVE READ ACCESS ONLY>>

     ITEMS:
                        ADDRESS,         2X30; <<2 LINES OF ADDRESS ALLOWED>>
                        CITY,             X16;
                        COUNTRY,          X12;
                        DATE,             I; <<PATH FOR ORDER-DATE, SHIP-DATE>>
                        NAME,             X30;
                        OPTION-DESC,      X10;
                        OPTION-PRICE,     L;
                        OPTION-TYPE,      I;
                        ORDER-DATE,       I;       <<MUST BE YYMM>>
                        ORDER-NO,         X10;
                        PRICE,            L;
                        PRODUCT-NO,       I;
                        PROD-DESC,        X30;
                        REGION,           X6;
                        REGION-DESC,      X30;
                        REGION-TYPE,      I;
                        SALESPERSON,      X4;
                        SHIP-DATE,        I;       <<MUST BE YYMM>>
                        STATE,            X6;
                        ZIP-CODE,         X8;

     SETS:

             NAME:      DATE,AUTOMATIC(3/10,15),SALES;
             ENTRY:     DATE(2);
             CAPACITY:  51;


             NAME:      ORDER,A(3/10,15);
             ENTRY:     ORDER-NO(2);
             CAPACITY:  101;


             NAME:      PRODUCT,MANUAL(3,10/15),SALES;
             ENTRY:     PRODUCT-NO(1),
                        PROD-DESC;
             CAPACITY:  11;


             NAME:      LOCATION,M(3,10/15),SALES;
             ENTRY:     REGION(1),
                        REGION-DESC,
                        REGION-TYPE;
             CAPACITY:  17;

$PAGE
```



```
PAGE  2                 HP250..6.A   EDITOR

55                       NAME:       OPTION,D(3/10,15);
56                       ENTRY:      ORDER-NO(ORDER),
57                                   OPTION-DESC,
57.1                                 OPTION-PRICE,
57.2                                 OPTION-TYPE;
58                       CAPACITY:   300;
59
60
61                       NAME:       CUSTOMER,DETAIL(3/10,15);
62                       ENTRY:      ORDER-NO(ORDER),
63                                   NAME,
64                                   ADDRESS,
65                                   CITY,
66                                   STATE,
67                                   COUNTRY,
68                                   ZIP-CODE,
69                                   ORDER-DATE(DATE),
70                                   SHIP-DATE(DATE),
71                                   REGION(LOCATION),
72                                   PRODUCT-NO(PRODUCT),
73                                   PRICE,
74                                   SALESPERSON;
75                       CAPACITY:   100;
76
77                       END.
```

APPENDIX B

SORT/250 Error Codes

211 **No DBASE IS statement active or bad data base specifier.** Attempt to execute an IN DATA SET or WORKFILE IS # without previously executing a DBASE IS or the data base that the DBASE IS was executed for has been closed. Or bad data base specified in DBASE IS.

212 **Specified data set not found.** An improper set name or number was specified.

230 **Improper nesting of SORT/250 statement.** An attempt was made to execute a SORT BY, FIND, IN DATA SET, DBASE IS, etc. while nested inside one of these statements. This can only happen if an expression uses a multi-line function subprogram.

231 **Cannot reactivate workfile.** An attempt is made to reactivate a workfile by using the WORKFILE IS # statement with no thread list, but the spcified file is not a workfile.

232 **Improper mode for SORT BY.** One of the data sets in the thread is not locked.

233 **No read access to specified data set, or data set not currently mounted.** One of the data sets in the thread is not accessible with the current password or is not mounted.

234 **Missing or improper data set linkage.** For WORKFILE IS #, two adjacent sets in the thread list have no path between them, or the chain id specified does not refer to an existing chain.

235 **No WORKFILE IS # statement active.** Attempt to execute a SORT BY or FIND when no workfile has been declared or the workfile was closed (either by de-assigning it or by DBCLOSE).

236 **Improper data item or data item not found.** The item specified in the LINK parameter of WORKFILE IS # does not refer to an item for the specified set or the given item in the SORT BY list is not linked via IN DATA SET to an item in one of the sets in the thread.

237 **Work record for sorting exceeds 256 bytes.** An attempt was made to issue a SORT BY where the sum of the length of the sort fields plus two times the thread length exceeded 256.

238 **Improper synthetic linkage.** The item in the LINK parameter of WORKFILE IS # either does not match the type of the search item in the master set following the LINK or it is not the first sub-item. Also, LINK is applied to a master set, or the set following the LINK is not of type master.

239 Insufficient space in workfile. The size of the workfile is insufficient to perform the desired operation.

240 Program lost due to disc failure. A disc error occurred when trying to re-load user memory from the workfile after completing a SORT BY. This will cause the system to execute SCRATCH A.

241 Improper operation attempted on workfile. Attempt to position the word pointer of a workfile to someplace other than word 1. Also, attempt to print an array on a workfile.

242 Improper READ # or PRINT # on workfile. A complete logical record was not read or written. The word pointer is reset to word 1.

243 Workfile contains invalid information. Attempt to access the workfile via SORT BY, FIND, READ # or PRINT # after its contents have been destroyed by a disc error or SHIFT HALT stopping a FIND or SORT.

245 SORT not allowed. INP Controller in use.

APPENDIX C

Determining Workfile Size

A facility is provided to determine the length of a workfile. This is accomplished by using the WORK program supplied on the Operating System Disc. To run the WORK program, execute:

```
RUN "WORK" ⇧
```

Two options are provided by this program, one for calculating a precise maximum size for specific values and one for plotting workfile size over a range. Press the appropriate softkey (or the corresponding keyboard SFK) to run the required function.

If EXACT FILE SIZE is selected, the following sets of prompts appear on the display. Simple ENTER the requested information (a,b,c and d) and the program will return a size for the workfile in sectors (256 byte records):

```
NUMBER OF SETS IN THREAD? a

NUMBER OF SORT FIELDS? b

NUMBER OF BYTES IN SORT FIELDS? c

NUMBER OF RECORDS TO SORT? d
```

After entering the requested information, the following summary is displayed:

```
THREAD LENGTH: a

NUMBER OF SORT FIELDS: b

NUMBER OF BYTES IN SORT FIELDS: c

NUMBER OF RECORDS TO SORT: d

NUMBER OF RECORDS REQUIRED: x
```

X represents the value returned for the minimum length workfile, in 256-byte records, needed to perform the specified sort.

Press EXIT to return to the main menu.

If SIZE GRAPH is selected, five prompts will appear on the display:

```
NUMBER OF SETS IN THREAD? a
NUMBER OF SORT FIELDS? b
NUMBER OF BYTES IN SORT FIELDS? c
MAX NUMBER OF RECORDS TO SORT? d
MIN NUMBER OF RECORDS TO SORT? e
```

After entering the required data, a summary listing is displayed. Press the PROCEED softkey to plot a graph showing the workfile size as a function of the number of records sorted. Notice that the graph is a non-linear function. Press EXIT to return to the main menu.

Four possible symbols are used to plot the graph and each one has a unique meaning described as follows:

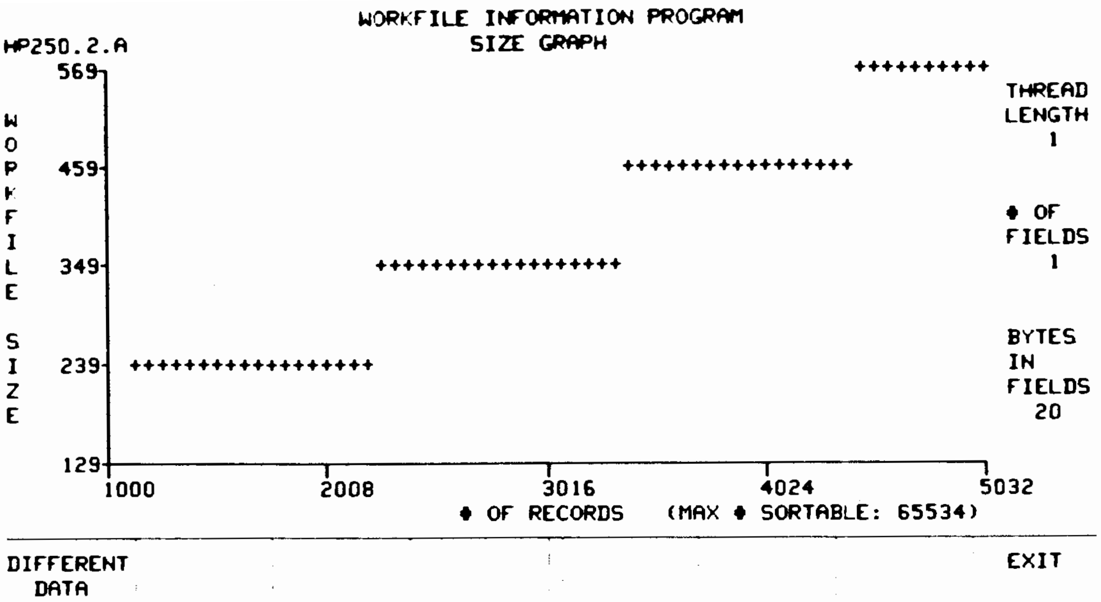
· - Sort can be accomplished in memory.
+ - One-pass merge required.
✚ - Two-pass merge required.
▒ - Non-processable sort. Requires greater than 65534 sectors of work space.

Two sample graphs are shown next, each for a thread length of 1 with one sort field. The first graph shows a sort field length of 20 bytes, while the second shows a length of 200 bytes.

Press HALT anytime during program execution to exit the WORK program.

NOTE

The WORK program assumes that a sort is to be performed. If only FINDs are performed, the workfile size is MAX ((T*L+127) DIV 128, 1), where T is the thread length and L is the number of entries found.

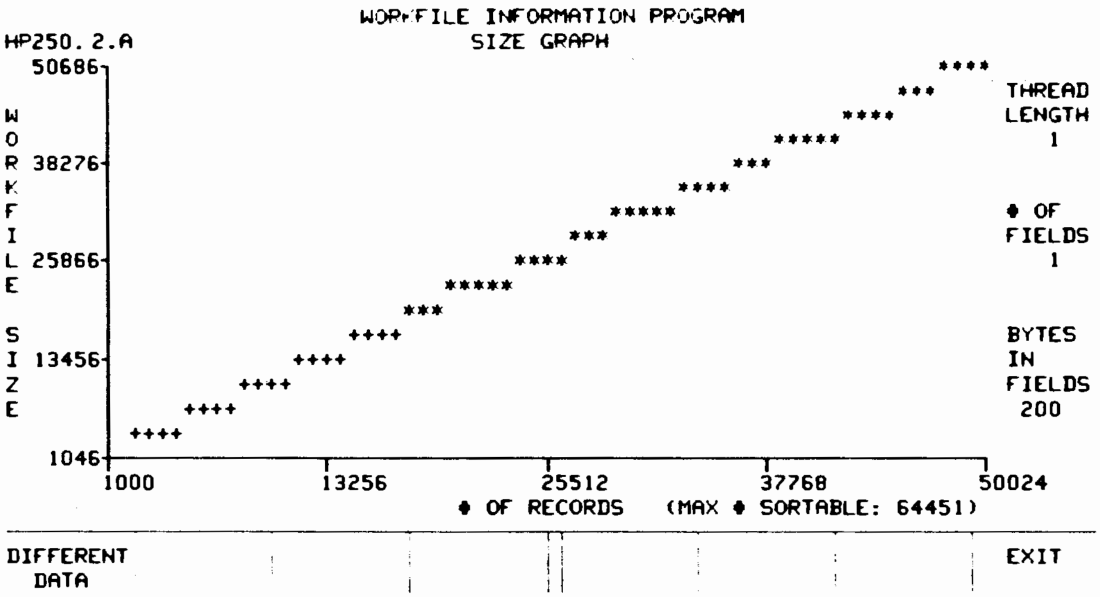


```
                         WORKFILE INFORMATION PROGRAM
HP250.2.A                          SIZE GRAPH
```

THREAD LENGTH 1

OF FIELDS 1

BYTES IN FIELDS 20



```
                         WORKFILE INFORMATION PROGRAM
HP250. 2.A                         SIZE GRAPH
```

THREAD LENGTH 1

OF FIELDS 1

BYTES IN FIELDS 200

Sort Field Size

The number of bytes in the sort fields is the sum of all the sort field lengths. These are:

Field Type	Length in Bytes
INTEGER	2
SHORT	4
REAL	8
STRING	length in characters

With certain local-language keyboards, however, there is some extra overhead because of the way sorting is done. For French, Italian and Spanish systems, four bytes per string should be added. For German systems, the effective length of each string should be computed via the formula:

$$\text{Actual Length} = L + 2*((L + 31)\text{DIV } 32)$$

where L = length of string in characters

Workfile Size Function

The WORK program defines a function to generate the graphs shown in Appendix D. Refer to the following listing for operational detail.

```
1000 !
1010 !   FUNCTION TO COMPUTE NUMBER OF RECORDS NEEDED
1020 !
1030 !   ENTRY: Thread_len is a number from 1 to 10 specifying the number
1040 !                   of sets in the thread.
1050 !          Wrsz is an even number between 2 and 256-2*Thread_len which
1060 !                   specifies the sum of the lengths of the sort keys
1070 !                   in bytes.
1080 !          Num_keys is a number between 1 and 10 specifying the number
1090 !                   of sort keys.
1100 !          Num_recs is a number between 0 and 65534 specifying the
1110 !                   maximum number of records to be sorted.
1120 !
1130 !   EXIT: The value returned is the number of sectors (256 byte logical
1140 !         records) needed to perform the indicated sort.  If a -1 is
1150 !         returned it means that one of the parameters was out of range.
1160 !         If the returned value exceeds 65534, the sort will be unprocess-
1170 !         able, since 250 files cannot contain more than 65534 sectors.
1180 !
1190     DEF FNWf_len(Thread_len,Wrsz,Num_keys,Num_recs)
1200       ! Convert Wrsz (in bytes) to Wr_size (in words).
1210       Wr_size=Wrsz/2
1220       !
1230       ! Test parameter ranges
1240       Test=FRACT(Thread_len)+FRACT(Wr_size)+FRACT(Numkeys)+FRACT(Num_recs)
1250       Test=Iest OR ((Thread_len(1) OR (Thread_len)10))
1260       Test=Test OR ((Wr_size(1) OR (Wr_size+Thread_len)128))
1270       Test=Test OR ((Num_keys(1) OR (Num_keys)10))
1280       Test=Test OR ((Num_recs(0) OR (Num_recs)65534))
1290       IF Test THEN
1300         !
1310         ! Define Wf_len as -1 since some of the parameters were bad.
1320         Wf_len=-1
1330       ELSE
1340         !
1350         ! Define the number of sectors needed to store user memory.
1360         Ps=121
1370         ! Calculate memory available for sorted records.
1380         Ms=13388-10*Thread_len-4*Num_keys
1390         ! Calculate maximum number of records that can be sorted in memory.
1400         Rim=Ms DIV (Wr_size+Thread_len+1)
1410         ! Calculate number of sectors needed to store pointers.
1420         Psec=(Thread_len*Num_recs+127) DIV 128
1430         IF Num_recs(Rim THEN
1440           !
1450           ! Define workfile size for case where the sort can be
1460           ! accomplished in memory with no need for merges.
1470           Wf_len=Ps+MAX(Psec,1)
1480         ELSE
1490           !
1500           ! Set up parameters for sort merge.
1510           !
1520           ! Define the magic number.
1530           K11m2=133*11-2
1540           ! Make guesses concerning when merges will be required.
1550           ! These guesses are necessarily inaccurate since for
1560           ! thread lengths greater than 2, SORT/250 really has
1570           ! no way of predicting the number of records to be sorted.
1580           Maxpos=MAX(32767*(Thread_len)2),(Num_recs+1) DIV 2)
1590           Maxsseg=(K11m2-2*(Maxpos DIV Rim)) DIV 10
1600           ! Determine how many work records can a put in one sector.
```

```
1610        Bf=128 DIV (Wr_size+Inread_len)
1620        ! Determine how many sectors are needed to dump a memory full
1630        ! of work records.
1640        Rpb=(Rim-1) DIV Bf+1
1650        ! Calculate space needed if miscellaneous records in memory must
1660        ! be dumped before the last pass (or only) pass merge can be
1670        ! performed.
1680        Mslop=(Num_recs MOD Rim-1) DIV Bf+1
1690        IF Num_recs<Maxsseg*Rim THEN
1700          !
1710          ! Define workfile size for case where only a one pass merge
1720          ! is required to accomplish the sort.
1730          Wf_len=Ps+Psec+Num_recs DIV Rim*Rpb+Mslop*(Num_recs DIV Rim)13)
1740        ELSE
1750          !
1760          ! Define workfile size for case where one or more intermediate
1770          ! merges are required to accomplish the sort.
1780          !
1790          ! Determine how many sectors are needed to store the results
1800          ! of an intermediate merge.
1810          Mrpb=(11*Rim-1) DIV Bf+1
1820          ! Calculate the amount of slop left over after the last merge.
1830          Xtra=Mslop*((Num_recs/Rim-Maxsseg) MOD 11)10)
1840          ! Determine how many intermediate merges will be performed.
1850          Lsegs=(Num_recs/Rim-Maxsseg) DIV 11+1
1860          ! AT LAST!  Define the required size.
1870          Wf_len=Ps+Psec+Maxsseg*Rpb+Xtra+Mrpb*Lsegs
1880        END IF
1890      END IF
1900    END IF
1910    RETURN Wf_len
1920  FNEND
```

Examples of SORT/250 Performance

This appendix provides the user with an intuitive "feel" for the time taken by a certain class of common operations. The data presented here is by no means intended to be comprehensive.

The following graphs compare number of records sorted versus sort time. The particular data base used was one constructed specifically for producing these graphs. There is one graph for each type of disc available with the HP250 system. The lines on each graph correspond to different sort field lengths.

The data base consists of a single stand-alone detail data set whose media record length is 100 bytes. The workfile length is 2000 sectors (no attempt was made to compute an optimal workfile size). The last graph shows the resulting minimum workfile sizes.

In the case of the flexible disc, the workfile and data base were on the same volume. In this latter case, however, no attempt was made to control the relative placement of the data set and the workfile. The workfile was empty at the start of each sort (REC=0).

Note that there are jumps in the time function for a particular field length. These jumps occur at the same place in the sort time function as in the workfile size function (see Appendix C). Since jumps occur everytime user memory is filled with records, the contents of memory must be written to the workfile before sorting can continue. The location of these jumps can be determined as follows:

Let:

I = number of sort fields
B = Sort field length (sum of all sort fields)
T = number of sets in the thread

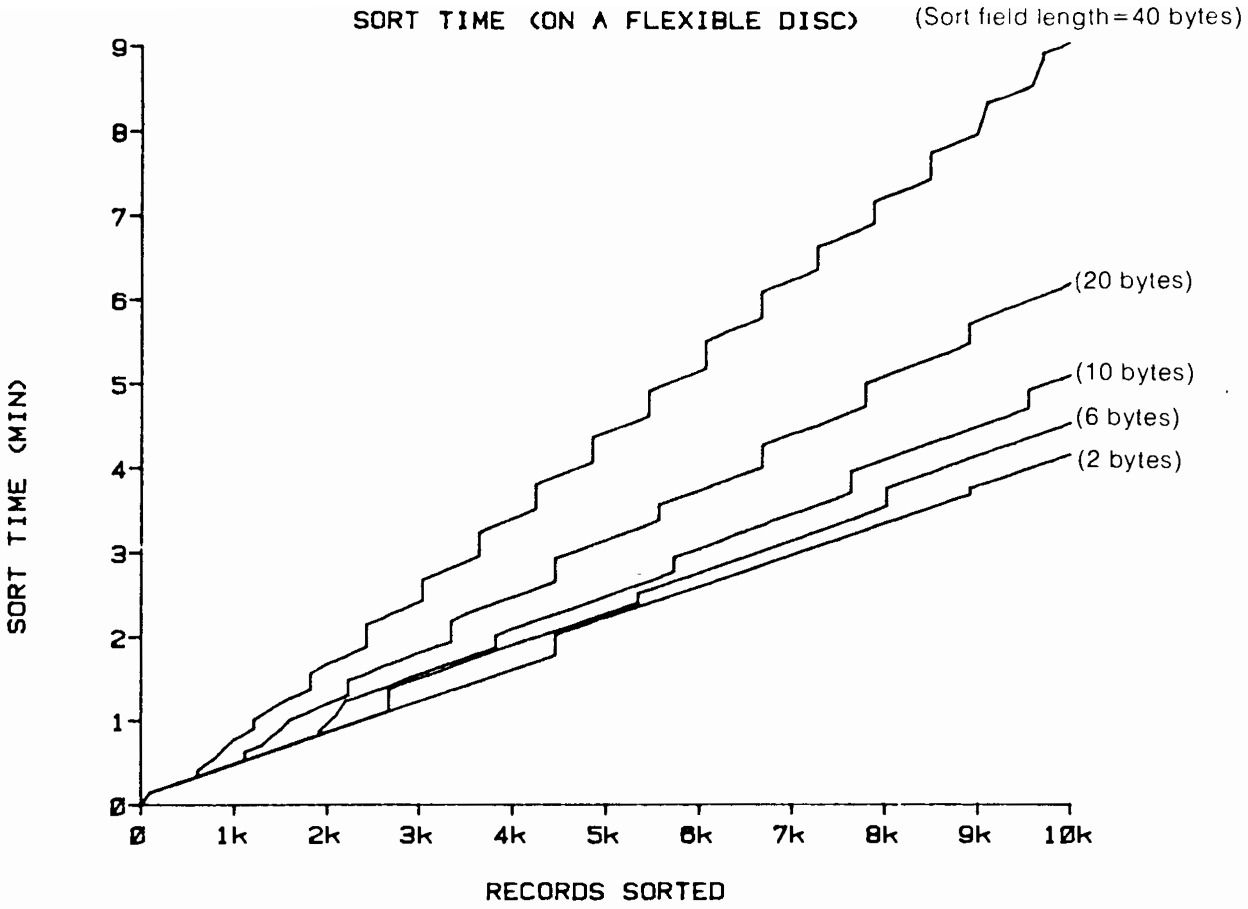
Then define:

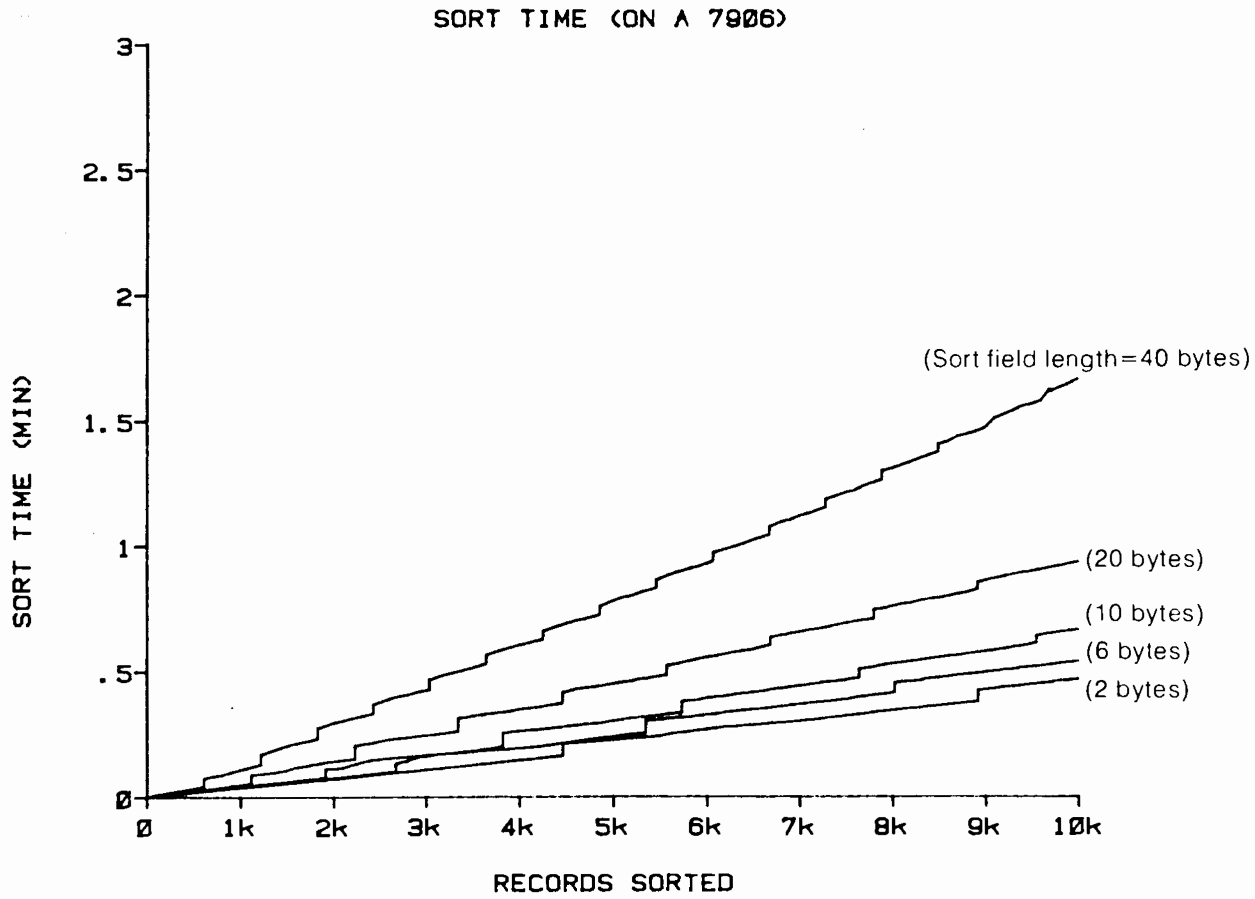
M = (26776−20*T−8*I) DIV (B+2*T+2)

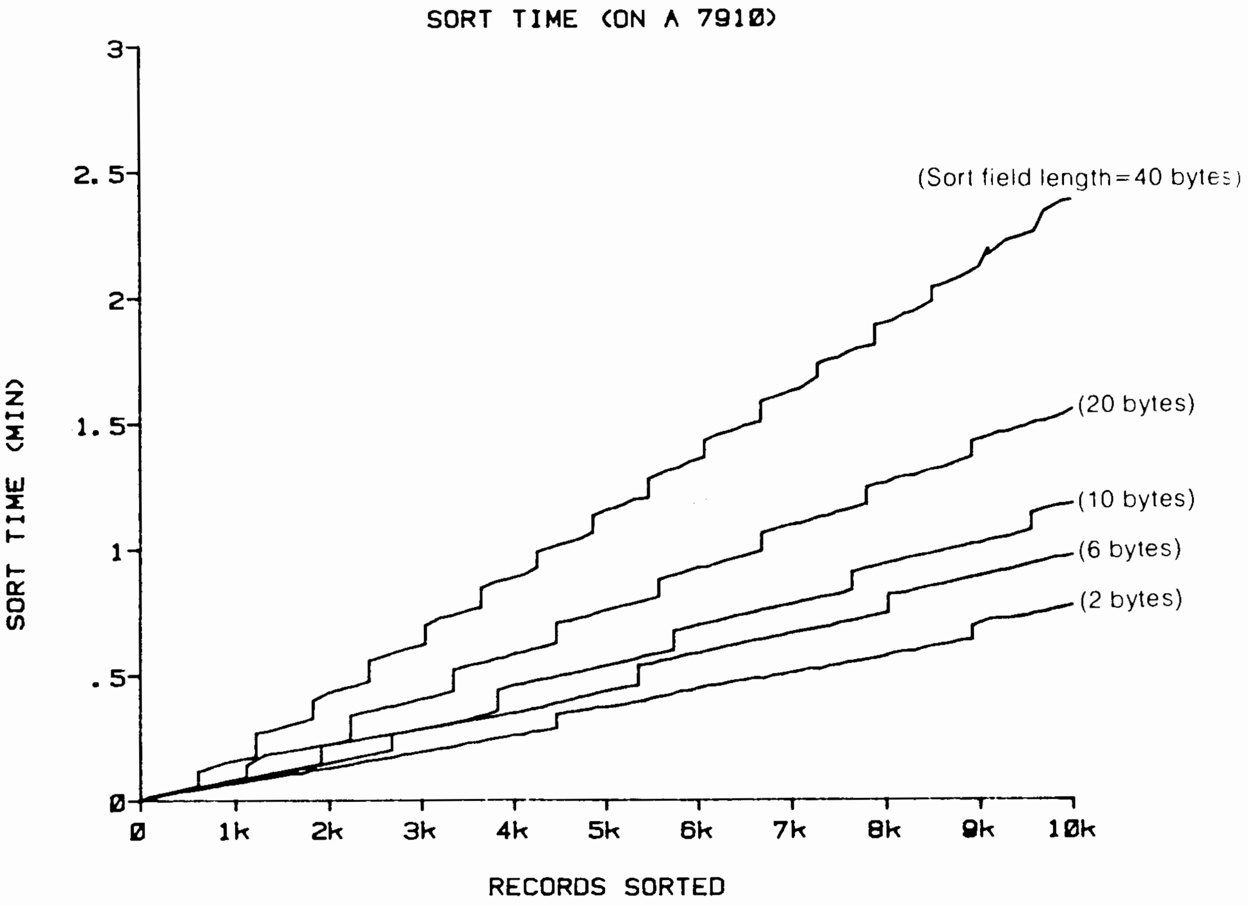
Jumps occur at:

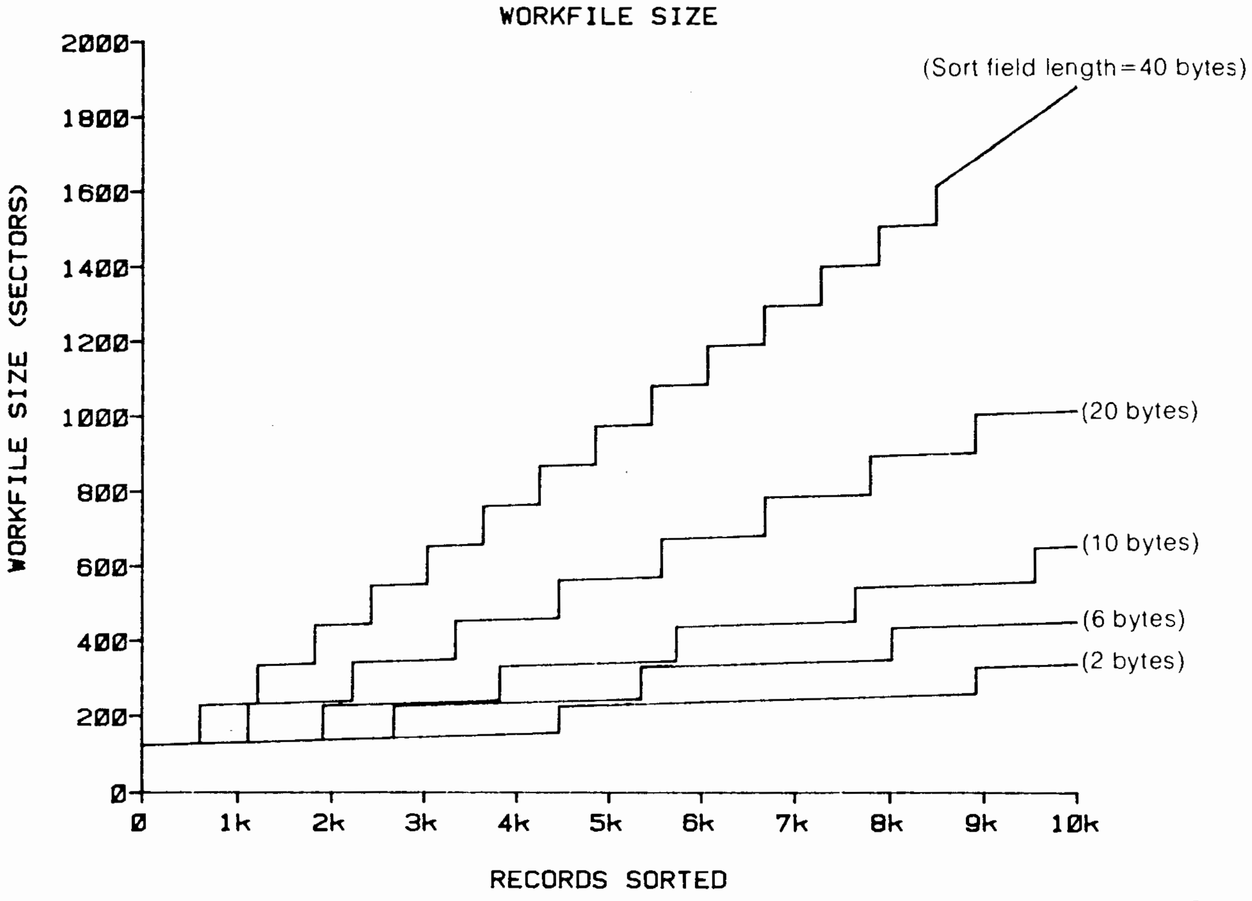
M, 2M, 3M, ..., 13M

As the multiplier or M exceeds 13, however, the jumps move down slightly.


SORT TIME (ON A FLEXIBLE DISC)
(Sort field length=40 bytes)
(20 bytes)
(10 bytes)
(6 bytes)
(2 bytes)
SORT TIME (MIN)
Ø
1
2
3
4
5
6
7
8
9
Ø
1k
2k
3k
4k
5k
6k
7k
8k
9k
1Øk
RECORDS SORTED


SORT TIME (ON A 79Ø6)
(Sort field length=40 bytes)
(20 bytes)
(10 bytes)
(6 bytes)
(2 bytes)
SORT TIME (MIN)
Ø
.5
1
1.5
2
2.5
3
Ø
1k
2k
3k
4k
5k
6k
7k
8k
9k
1Øk
RECORDS SORTED


SORT TIME (ON A 791Ø)
SORT TIME (MIN)
3
2.5
2
1.5
1
.5
Ø
Ø
1k
2k
3k
4k
5k
6k
7k
8k
9k
1Øk
RECORDS SORTED
(Sort field length=40 bytes)
(20 bytes)
(10 bytes)
(6 bytes)
(2 bytes)


WORKFILE SIZE
WORKFILE SIZE (SECTORS)
2ØØØ
18ØØ
16ØØ
14ØØ
12ØØ
1ØØØ
8ØØ
6ØØ
4ØØ
2ØØ
Ø
Ø
1k
2k
3k
4k
5k
6k
7k
8k
9k
1Øk
RECORDS SORTED
(Sort field length=40 bytes)
(20 bytes)
(10 bytes)
(6 bytes)
(2 bytes)

HEWLETT
PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Tecnica de
Equipamentos
Eléctricos S A R L
R Barbosa Rodrigues, 42-I DT
Caixa Postal 6487
Luanda
Tel 35515 6
Cable TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty Ltd
31-41 Joseph Street
Blackburn, Victoria 3130
P O Box 36
Doncaster East Victoria 3109
Tel 89-6351
Telex 31-024
Cable HEWPARD Melbourne

Hewlett-Packard Australia
Pty Ltd
31 Bridge Street
Pymble
New South Wales, 2073
Tel 449-6566
Telex 21561
Cable HEWPARD Sydney

Hewlett-Packard Australia
Pty Ltd
153 Greenhill Road
Parkside S A . 5063
Tel 272-5911
Telex 82536 ADEL
Cable HEWPARD ADELAID

Hewlett-Packard Australia
Pty Ltd
141 Stirling Highway
Nedlands W A 6009
Tel 86-5455
Telex 93859 PERTH
Cable HEWPARD PERTH

Hewlett-Packard Australia
Pty Ltd
121 Wollongong Street
Fyshwick A C T 2609
Tel 95-2733
Telex 62650 Canberra
Cable HEWPARD CANBERRA

Hewlett Packard Australia
Pty Ltd
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill 4000 Queensland
Tel 229-1544
Cable HEWPARD Brisbane

GUAM
Medical Pocket Calculators Only
Guam Medical Supply Inc
Jay Ease Building Room 210
P O Box 8947
Tamuning 96911
Tel 646-4513
Cable EARMED Guam

HONG KONG
Schmidt & Co (Hong Kong) Ltd
P O Box 297
Connaught Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel H 255291-5
Telex 74766 SCHMC HX
Cable SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel 29 50 21
Telex 001-2156
Cable BLUEFROST

Blue Star Ltd
Sahas
414 2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel 45 78 87
Telex 011-4093
Cable FROSTBLUE

Blue Star Ltd
Band Box House
Prabhadevi
Bombay 400 025
Tel 45 73 01
Telex 011-3751
Cable BLUESTAR

Blue Star Ltd
7 Hare Street
P O Box 506
Calcutta 700 001
Tel 23-0131
Telex 021-7655
Cable BLUESTAR

Blue Star Ltd
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel 634770 & 635166
Telex 031-2463
Cable BLUESTAR

Blue Star Ltd
Blue Star House
11 11A Magarath Road
Bangalore 560 025
Tel 55668
Telex 043-430
Cable BLUESTAR

Blue Star Ltd
Meeakshi Mandiran
xxx 1678 Mahatma Gandhi Rd
Cochin 682 016
Tel 32069.32161.32282
Telex 0885-514
Cable BLUESTAR

Blue Star Ltd
1-1-117 1
Sarojini Devi Road
Secunderabad 500 003
Tel 70126. 70127
Cable BLUEFROST
Telex 015-459

Blue Star Ltd
2 34 Kodambakkam High Road
Madras 600034
Tel 82056
Telex 041-379
Cable BLUESTAR

INDONESIA
BERCA Indonesia P T
P O Box 496 Jkt
JLNeAbdul Muis 62
Jakarta
Tel 40369 49886 49255.356038
JKT 42895
Cable BERCACON

BERCA Indonesia P 1
63 JL Raya Gubeng
Surabaya
Tel 44309

ISRAEL
Electronics & Engineering Div
of Motorola Israel Ltd
17 Kremenetski Street
P O Box 25016
Tel-Aviv
Tel 38973
Telex 33569
Cable BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku. **Tokyo** 151
Tel 03-370-2281/92
Telex 232-2024YHP MARKET
TOK 23-724
Cable YHPMARKET

Yokogawa-Hewlett-Packard Ltd
Chuo Bldg 4th Floor
4-20 Nishinakajima 5-chome
Yodogawa-ku Osaka-shi
Osaka 532
Tel 06-304-6021

Yokogawa-Hewlett-Packard Ltd
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku. **Nagoya** 450
Tel (052) 571-5171

Yokogawa-Hewlett-Packard Ltd
Tanigawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel 045-312-1252
Telex 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd
Mito Mitsu Building
105 Chome-1.San-no-maru
Mito Ibaragi 310
Tel 0292-25-7470

Yokogawa-Hewlett-Packard Ltd
Inoue Building
1348-3 Asahi-cho 1-chome
Atsugi, Kanagawa 243
Tel 0462-24-0452

Yokogawa-Hewlett-Packard Ltd
Kumagaya Asahi
Hackijuni Building
4th Floor
3-4 Tsukuba
Kumagaya Saitama 360
Tel 0485-24-6563

KENYA
Technical Engineering
Services(E A)Ltd
P O Box 18311
Nairobi
Tel 557726 556762
Cable PROTON

Medical Only
International Aeradio(E A)Ltd
P O Box 19012
Nairobi Airport
Nairobi
Tel 336055 56
Telex 22201 22301
Cable INTAERIO Nairobi

KOREA
Samsung Electronics Co . Ltd
20th Fl Dongbang Bldg 250. 2-KA
C P O Box 2775
Taepyung-Ro. Chung-Ku
Seoul
Tel (23) 6811
Telex 22575
Cable ELEKSTAR Seoul

MALAYSIA
Teknik Mutu Sdn Bhd
2 Lorong 13 6A
Section 13
Petaling Jaya **Selangor**
Tel 54994 54916
Telex MA 37605

Protel Engineering
P O Box 1917
Lot 259. Satok Road
Kuching **Sarawak**
Tel 2400
Cable PROTEL ENG

MOZAMBIQUE
A N Goncalves Lta
162 1 Apt 14 Av D Luis
Caixa Postal 107
Lourenco Marques
Tel 27091. 27114
Telex 6-203 NEGON Mo
Cable NEGON

NEW ZEALAND
Hewlett-Packard (N Z) Ltd
P D Box 9443
Courtenay Place
Wellington
Tel 877-199
Cable HEWPACK Wellington

Hewlett-Packard (N Z) Ltd
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel 569-651
Cable HEWPACK.Auckland

Analytical Medical Only
Medical Supplies N Z Ltd
Scientific Division
79 Carlton Gore Rd . Newmarket
P O Box 1234
Auckland
Tel 75-289
Cable DENTAL Auckland

Analytical Medical Only
Medical Supplies N Z Ltd
P O Box 1994
147-161 Tory St
Wellington
Tel 850 799
Telex 3858
Cable DENTAL. Wellington

Analytical Medical Only
Medical Supplies N Z Ltd
P O Box 309
239 Stanmore Road
Christchurch
Tel 892-019
Cable DENTAL Christchurch

Analytical Medical Only
Medical Supplies N Z Ltd
303 Great King Street
P O Box 233
Dunedin
Tel 88-817
Cable DENTAL Dunedin

NIGERIA
The Electronics
Instrumentations Ltd
N6B 770 Oyo Road
Oluseun House
P M B 5402
Ibadan
Tel 61577
Telex 31231 TEIL Nigeria
Cable THETEIL Ibagan

The Electronics Instrumentations Ltd
144 Agege Motor Road Mushin
P O Box 6645
Lagos
Cable THETEIL Lagos

PAKISTAN
Mushko & Company. Ltd
Oosman Chambers
Abdullah Haroon Road
Karachi 3
Tel 511027 512927
Telex 2894
Cable COOPERATOR Karachi

Mushko & Company Ltd
38B Satellite Town
Rawalpindi
Tel 41924
Cable FEMUS Rawalpindi

PHILIPPINES
The Online Advanced
Systems Corporation
Rico House
Amorsolo cor Herrera Str
Legaspi Village. Makati
Metro **Manila**
Tel 85-35-81 85-34-91
Telex 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P O Box 3458
Salisbury
Tel 705231 (5 lines)
Telex RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte) Ltd
1150 Depot Road
Alexandra P O Box 58
Singapore 4
Tel 270-2355
Telex HPSG RS 21486
Cable HEWPACK Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty) Ltd
Private Bag Wendywood
Sandton Transvaal 2144
Hewlett-Packard Centre
Daphne Street Wendywood
Sandton Transvaal 2144
Tel 802-10408
Telex 8-4782
Cable HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa
(Pty) Ltd
P O Box 39325
Gramley Sandton 2018
451 Wynberg Extension 3
Sandton, 2001
Tel 636-8188 9
Telex 8-2391

Hewlett-Packard South Africa
(Pty). Ltd
P O Box 120
Howard Place Cape Province 7450
Pine Park Centre. Forest Drive
Pinelands Cape Province 7405
Tel 53 7955 thu 9
Telex 57-0006

Service Department
Hewlett-Packard South Africa
(Pty) Ltd
P O Box 37099
Overport Durban 4067
Braby House
641 Ridge Road
Durban 4001
Tel 88-7478
Telex 6-7954

TAIWAN
Hewlett-Packard Far East Ltd
Taiwan Branch
39 Chung Hsiao West Road
Sec 1 7th Floor
Taipei
Tel 3819160-4
Cable HEWPACK TAIPEI

Hewlett-Packard Far East Ltd
Taiwan Branch
68-2 Chung Cheng 3rd Road
Kaohsiung
Tel (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co Ltd
No 20 Yung Sui Road
Taipei
Tel 3715171-4 (5 lines)
Telex 22894 SANKWANG
Cable SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E A) Ltd
P O Box 861
Dar es Salaam
Tel 21251 Ext 265
Telex 41030

THAILAND
UNIMESA Co Ltd
Elcom Research Building
2538 Sukumvit Ave
Bangkok
Tel 3932387 3930338
Cable UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E A) Ltd
P O Box 2577
Kampala
Tel 54388
Cable INTAERIO Kampala

ZAMBIA
R J Tilbury (Zambia) Ltd
P O Box 2792
Lusaka
Tel 73793
Cable ARJAYTEE. Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto California 94304
Tel (415) 493-1501
TWX 910-373-1267
Cable HEWPACK Palo Alto

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd
11620A - 168th Street
EdmontonT5M 3T9
Tel (403) 452-3670
TWX 610-831-2431

Hewlett-Packard (Canada) Ltd
210.7220 Fisher St S E
Calgary T2H 2H8
Tel (403) 253-2713
Twx 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd
837 E Cordova Street
Vancouver V6A 3R2
Tel (604) 254-0531
TWX 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd
513 Century St
St James
Winnipeg R3H DL8
Tel (204) 786-7581
TWX 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd
800 Windmill Road
Dartmouth B3B 1L1
Tel (902) 469-7820
TWX 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd
1020 Morrison Dr
Ottawa K2H 8K7
Tel (613) 820-6483
TWX 610-563-1636

Hewlett-Packard (Canada) Ltd
6877 Goreway Drive
Mississauga L4V 1M8
Tel (416) 678-9430
TWX 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd
275 Hymus Blvd
Pointe Claire H9R 1G7
Tel (514) 697-4232
TWX 610-422-3022
TLX 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd in Mississauga

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S A
Av Leandro N Alem 822 - 12
1001**Buenos Aires**
Tel 31-6063.4.5.6 and 7
Telex 122443 AR CIGY
Cable HEWPACK ARG

BOLIVIA
Casa Kavlin S A
Calle Potosi 1130
P O Box 500
La Paz
Tel 41530.53221
Telex CWC BX 5298.ITT 3560082
Cable KAVLIN

BRAZIL
Hewlett-Packard do Brasil
I e C Ltda
Avenida Rio Negro 980
Alphaville
06400**Barueri** SP
Tel 429-3222

Hewlett-Packard do Brasil
I e C Ltda
Rua Padre Chagas 32
90000-**Porto Alegre**-RS
Tel (0512) 22-2998. 22-5621
Cable HEWPACK Potto Alegre

Hewlett-Packard do Brasil
I E C Ltda
Rua Siqueira Campos. 53
Copacabana
20000-**Rio de Janeiro**
Tel 257-80-94-ODD (021)
Telex 391-212-1905 HEWP-BR
Cable HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda
Alameda 580-Of 807
Casilla 2118
Santiago, 1
Tel 398613
Telex 3520001 CALMET
Cable CALMET Santiago

COLOMBIA
Instrumentacidn
Henrik A Langebaek & Kier S A
Carrera 7 No 48-75
Apartado Aéreo 6287
Bogota, I D E
Tel 69-88-77
Cable AARIS Bogota
Telex 044-400

COSTA RICA
Cientifica Costarricense S A
Avenida 2 Calle 5
San Pedro de Montes de Oca
Apartado 10159
San Jose
Tel 24-38-20 24-08-19
Telex 2367 GALGUR CR
Cable GALGUR

ECUADOR
Calculators Only
Computadoras y Equipos
Electrdnicos
P D Box 6423 CCI
Eloy Alfaro #1824.3 Piso
Quito
Tel 453482
Telex 02-2113 Sagita Ed
Cable Sagita-Duito

EL SALVADOR
Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes 11-48
San Salvador
Tel 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48
Zona 9
Guatemala City
Tel 63627 64786
Telex 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana
S A de C V
Av Periferico Sur No 6501
Tepepan. Xochimilco
Mexico 23. D F
Tel 905-676-4600

Hewlett-Packard Mexicana.
S A de C V
Ave Constitucidn No 2184
Monterrey N L
Tel 48-71-32 48-71-84
Telex 038-410

NICARAGUA
Roberto Teran G
Apartado Postal 689
Edificio Terán
Managua
Tel 25114. 23412.23454
Cable ROTERAN Managua

PANAMA
Electrdnico Balboa. S A
P D Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel 64-2700
Telex 3483103 Curunda.
Canal Zone
Cable ELECTRON Panama

PERU
Compañia Electro Médica S A
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel 41-4325
Cable ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272
No 203 Urb Country Club
Carolina 00924
Tel (809) 762-7255
Telex 345 0514

URUGUAY
Pablo Ferrando S A
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel 40-3102
Cable RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C A
P O Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel 35 00 11 (20 lines)
Telex 25146 HEWPACK
Cable HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT
Hewlett-Packard
Inter-Americas
3200 Hillview Ave
Palo Alto California 94304
Tel (415) 493 1501
TWX 910-373-1260
Cable HEWPACK Palo Alto
Telex 034-8300 034-8493