

HP 3000 Computer Systems



SORT-MERGE/3000

Reference Manual



19447 PRUNERIDGE AVE., CUPERTINO, CALIFORNIA 95014

Part No. 32214-90002
Product No. 32214C
U0984

Printed in U.S.A. 9/84

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

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.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the most recent date on which the technical material on any given page was altered. If a page is simply re-arranged due to a technical change on a previous page, it is not listed as a changed page. Within the manual, changes are marked with a vertical bar in the margin.

First Edition	Sept 1981
Update 1	Sept 1984

Changed Pages	Effective Date
i to iv	Sept 1984
1-2	Sept 1984
2-2 to 2-3	Sept 1984
2-11	Sept 1984
2-13	Sept 1984
2-15 to 2-17	Sept 1984
2-19	Sept 1984
2-23	Sept 1984
2-27	Sept 1984
2-42	Sept 1984
2-47A	Sept 1984
2-49A	Sept 1984
3-2	Sept 1984
3-4 to 3-10	Sept 1984
3-31A to 3-32A	Sept 1984
4-2 to 4-4	Sept 1984
4-8	Sept 1984
4-18 to 4-18A	Sept 1984
5-2	Sept 1984
5-4 to 5-7	Sept 1984
5-9 to 5-10	Sept 1984
5-21 to 5-24	Sept 1984
6-2 to 6-4	Sept 1984
6-6	Sept 1984
6-8	Sept 1984
6-15	Sept 1984
6-17	Sept 1984
A-1 to A-3	Sept 1984
A-5 to A-7	Sept 1984
C-1	Sept 1984
Index-1 to Index-2	Sept 1984

PRINTING HISTORY

New editions incorporate all update material since the previous edition. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover changes only when a new edition is published. If minor corrections and updates are incorporated, the manual is reprinted but neither the date on the title page and back cover nor the edition change.

First Edition Sept 1981
Update 1 Sept 1984

This publication is the reference manual for SORT-MERGE/3000. SORT-MERGE/3000 is a subsystem of the MPE/3000 Operating System and consists of two programs: a Sort program and a Merge program. The SORT-MERGE/3000 subsystem sorts a file of records or merges multiple files of sorted records into a single file.

The Sort and Merge programs can be run as stand-alone programs controlled by direct user commands, or they can be called from user programs. Examples are provided throughout this manual which demonstrate how to run Sort and Merge as stand-alone programs and how to call them from SPL/3000 (Systems Programming Language for the HP 3000 Computer System) and FORTRAN/3000 (a version of FORTRAN IV for the HP 3000 Computer System). (The COBOL programmer uses the COBOL SORT or MERGE verb to run SORT-MERGE/3000).

The content of this publication is:

Section I

introduces the SORT-MERGE/3000 subsystem. The concepts of sorting and merging files are discussed, and the basic structure of SORT-MERGE/3000 is explained.

Section II

provides instructions for executing the SORT and MERGE programs as stand-alone programs. Examples are provided which demonstrate running the Sort program in interactive and batch modes.

Section III

explains how to call SORT intrinsics from FORTRAN/3000 programs. Also provided are definitions of the SORT intrinsics, and complete, operating programs in FORTRAN/3000.

Section IV

explains how to call MERGE intrinsics from FORTRAN/3000 programs. It also provides definitions of the MERGE intrinsics and FORTRAN example programs.

Section V

explains how to call SORT intrinsics from SPL/3000 programs. Operating programs are used as examples.

Section VI

explains how to call MERGE intrinsics from SPL/3000 programs. Operating programs are used as examples.

Appendix A

provides a list of error messages.

Appendix B

contains a table of ASCII and EBCDIC characters.

Index

contains an alphabetical listing of the main topics of the manual.

The following manuals are available for reference:

MPE Intrinsics Reference Manual (30000-90010)
MPE Commands Reference Manual (30000-90009)
FORTRAN Reference Manual (30000-90040)
System Programming Language Reference Manual (30000-90024)
COBOL/3000 Reference Manual (32213-90001)
COBOL II/3000 Reference Manual (32233-90001)
System Reference Manual (30000-90020)
Native Language Support Reference Manual (32414-90001)

CONVENTIONS USED IN THIS MANUAL

NOTATION	DESCRIPTION
[]	<p>An element inside brackets is optional. Several elements stacked inside a pair of brackets means the user may select any one or none of these elements.</p> <p>Example: $\begin{bmatrix} X \\ Y \end{bmatrix}$ user may select X or Y, or neither.</p>
{ }	<p>When several elements are stacked within braces the user must select one of these elements.</p> <p>Example: $\left\{ \begin{array}{l} string \\ num\ byte \\ range\ string \end{array} \right\}$</p>
Italics	<p>Italics in lowercase denote a parameter which should be replaced by a user-supplied variable.</p> <p>Example: OUTPUT <i>filename</i>, ,KEY</p>
Underlining	<p>Where it is necessary to distinguish user input from computer output, the input is underlined.</p> <p>Example: PURGE OLD OUTPUT FILE REST.PUB.SYS? <u>YES</u></p>
<i>return</i>	<p>Indicates a carriage return</p>
:	<p>Command identifier character</p>
Capital letters	<p>Command name or literal information (parameter) to be entered.</p> <p>:RUN SORT.PUB.SYS</p> <p>Example: or</p> <p>>KEY 1, 10, PACKED</p>
Commas	<p>Separate positional parameters</p> <p>Example: >INPUT R, , 85</p> <p>The omission of the second parameter is indicated by two successive commas.</p>

NOTATION**DESCRIPTION**

Semicolons Separate keyword parameters and key specifications in the **KEY** command.

:PREPRUN \$OLDPASS; MAXDATA=4000; LIB=G

Examples: or

>KEY 31, 14; 1, 15

Superscripts:

C	Control character
BA	Byte array
DV	Double integer by value
I	Integer by reference
IV	Integer by value
IA	Integer array
L	Logical by reference
LV	Logical by value
LA	Logical array
LP	Logical procedure
P	Procedure
O-V	Optional variable

TABLE OF CONTENTS

SECTION I	INTRODUCING SORT-MERGE/3000	
KEY		1-1
ORDERING SEQUENCE		1-2
USING SORT INTERACTIVELY		1-2
USING MERGE INTERACTIVELY		1-5
SECTION II	RUNNING SORT AND MERGE AS STAND-ALONE PROGRAMS	
FILE DEFINITIONS		2-3
ALTSEQ		2-4
DATA		2-7
END		2-9
EXIT		2-10
INPUT(sort)		2-11
INPUT(merge)		2-13
KEY		2-14
LANGUAGE		2-16A
OUTPUT(sort)		2-17
OUTPUT(merge)		2-19
RESET		2-21
SHOW		2-22
VERIFY		2-24
CONTROL Y		2-26
EOD		2-27
EXAMPLES		
TYPES OF MODIFICATION SPECIFICATIONS		2-28
SHOW COMMAND		2-34
SORT OPERATION		2-42
MERGE OPERATION		2-48
SECTION III	CALLING SORT FROM A FORTRAN/3000 PROGRAM	
SORT PROGRAM INTRINSICS		3-1
SORTINIT		3-4
SORTINPUT		3-8
SORTOUTPUT		3-9
SORTEND		3-10
SORTSTAT		3-11
SORTTITLE		3-12
SORTERRORMESS		3-13
SORTINITIALF		3-14
EXAMPLES		
SORTINIT when both inputfiles and outputfiles specified		3-15
SORTINIT with ALTSEQ parameter		3-20
SORTINIT with EBCDIC as the collating sequence		3-22
Using the KEYCOMPARE parameter		3-24
Using the ERRORPROC parameter without the occurrence of an error		3-25
Using the ERRORPROC parameter during the occurrence of an error		3-27
Displaying the SORT statistics with the statistics parameter		3-29
Calling SORTINPUT		3-30

Calling SORTOUTPUT	3-32
Calling SORTSTAT	3-33
Calling SORTTITLE	3-34
Calling SORTINITIALF with the failure parameter	3-36
Multirecord, NOBUF, and Buffered Files	3-37
SECTION IV CALLING MERGE FROM A FORTRAN/3000 PROGRAM	
MERGE PROGRAM INTRINSICS	4-1
MERGEINIT	4-3
MERGEOUTPUT	4-7
MERGEEND	4-8
MERGESTAT	4-9
MERGETITLE	4-10
MERGEERRORMESS	4-11
MERGE	4-12
EXAMPLES	4-13
Calling MERGEINIT with the preprocessor parameter	4-13
Using the KEYONLY parameter	4-15
Calling MERGEINIT with the STATISTICS parameter	4-17
Calling MERGEOUTPUT	4-18
Calling MERGESTAT	4-20
Calling MERGETITLE	4-21
Calling MERGEERRORMESS from the ERRORPROC subroutine ER	4-23
Calling MERGE with the FAILURE parameter	4-25
SECTION V CALLING SORT FROM A SPL/3000 PROGRAM	
SORT PROGRAM INTRINSICS	5-1
SORTINIT	5-4
SORTINPUT	5-8
SORTOUTPUT	5-9
SORTEND	5-10
SORTSTAT	5-11
SORTTITLE	5-12
SORTERRORMESS	5-13
SORTINITIAL	5-14
EXAMPLES	5-15
Calling SORTINIT with the ALTSEQ parameter	5-15
Calling SORTINITIAL and SORTINPUT	5-16
Calling SORTOUTPUT	5-18
Calling SORTINITIAL without the INPUTFILE and OUTPUTFILE parameters	5-19
SECTION VI CALLING MERGE FROM A SPL/3000 PROGRAM	
MERGE PROGRAM INTRINSICS	6-1
MERGEINIT	6-3
MERGEOUTPUT	6-7
MERGEEND	6-8
MERGESTAT	6-9
MERGETITLE	6-10
MERGEERRORMESS	6-11
MERGE	6-12

EXAMPLES	6-15
Merging files opened MR and NOBUF	6-15
APPENDIX A ERROR MESSAGES AND RECOVERY PROCEDURES	
SORT ERROR MESSAGES	A-1
ALTSEQ ERROR MESSAGES	A-4
MERGE ERROR MESSAGES	A-5
RECOVERY PROCEDURES	A-7
APPENDIX B ASCII/EBCDIC/HOLLERITH TABLE	B-1
■ APPENDIX C NATIVE LANGUAGE COLLATING	C-1

INTRODUCING SORT-MERGE/3000

SORT-MERGE/3000 is a subsystem of the HP 3000 Multiprogramming Executive (MPE) operating system that allows you to sort one or more files, or merge several sorted files, to form one file in a specified sequence. Consider the output file to be a transformation of the input file in which only the order of the records is changed. SORT-MERGE/3000 is useful at two different levels. It can be used as a stand alone utility in which you interactively specify the input and output files, the sorting (or merging) keys, and the collating sequence. You can also call it from a FORTRAN/3000 (Sections III-IV), an SPL/3000 (Sections V-VI), or a COBOL II/3000 program. The programmatic use of SORT-MERGE/3000 from a COBOL II/3000 program is described in the COBOL II/3000 reference manual.

This section is useful for the first time user who is not an experienced programmer. It describes some of the basic terms and concepts used by SORT-MERGE/3000 and shows how you can use the subsystem interactively in simple cases.

KEY

A key is a section of the record used by SORT-MERGE/3000 to determine the order in which records are rearranged in a file. It is a group of characters you specify by stating the position of the first character and the number of characters in the group. Use the KEY command (see Section II) to establish the keys. For example, the KEY command,

➤KEY 44, 12

means the key is a character string starting at the 44th column position of the record and is 12 characters long. Fig. 1-1 shows three records, each containing the last name, the first name, the occupation, and the year of birth. Positions 1 through 10 define the key in that the records are alphabetized by the last names. This is specified by the command ➤KEY 1, 10. If the occupation is a key (specified by ➤KEY 31, 10), the records are reordered with the third record preceding the second.

POSITIONS					
1	2	3	4	5	6
1234567890	1234567890	1234567890	1234567890	1234567890	123456789
CLIFT	MONTGOMERY	ACTOR	BORN	1920	<i>record 1</i>
JOPLIN	JANIS	SINGER	BORN	1943	<i>record 2</i>
VANDEBILT	CORNELIUS	CAPITALIST	BORN	1794	<i>record 3</i>
<i>characters starting at the 1st position</i>			<i>characters starting at the 31st position</i>		

Figure 1-1. Key Positions

Keys must appear in the same relative position in each record of a file. If you specify the first five positions of a record in a file as a key, the first five positions in every other record are considered as a key. The data format for all such keys must be the same. For the same reason, keys in the files merged must be in the same relative position and have the same data format. You can even specify more than one key (Fig. 1-3). In the case of multiple keys, the key you type in first is called the major key. SORT-MERGE/3000 uses the major key to rearrange the records. In the event of ties, the second key you type in determines the precedence of the records in the final form. If both the first and second keys have the same value, the third key is used, and so forth. If all the keyfields in two or more records are identical, the order of the input records is preserved in the output file.

ORDERING SEQUENCE

SORT-MERGE/3000 arranges records in a file according to the value of the data in the keys. The individual characters defining the keys determine these values based on their positions in a collating sequence. The collating sequence you choose may be ASCII, EBCDIC, a native language, or user defined. Appendix B shows the order of the ASCII collating sequence.

ASCII/EBCDIC. These are the basic collating sequences, assigned by the Data command (Section II). You can modify these sequences to define an alternative sequence with the ALTSEQ command (Section II). In most applications, the ASCII sequence is used for sorting and merging, although EBCDIC is used occasionally.

Native Language Collating. These are the collating sequences defined for the various native languages supported on the HP 3000. These apply to keys of type CHARACTER and are assigned by the LANGUAGE command. See Appendix C for further information on language collating.

Ascending/Descending. Records are considered in the ascending order if the key value of each record is greater than or equal to the preceding record according to the ASCII (or EBCDIC or a user defined) collating sequence. For example, the series, C, E, T, W, Z, is in ascending order. This is the order in which the records are compared unless you specify a descending order. If the key value of each record is less than or equal to the preceding record, the records are in the descending order. For example, 7, 3, 1, 0, is a descending order.

USING SORT INTERACTIVELY

Figs. 1-2 through 1-5 illustrate the use of the basic SORT-MERGE/3000 commands. However, these examples present only a small subset of the commands available to you as an interactive user. Note even for these few commands, some of the more involved options are omitted. You should consult Section II for a detailed description of all the commands. Fig. 1-2 takes an existing unsorted file, A (input file), and sorts it into a new file, AMERICAN (output file). A is displayed by using the EDIT/3000 subsystem. Your first step in the sorting procedure is telling MPE to run the SORT program by giving the following command:

```
:RUN SORT.PUB.SYS
```

› is a prompt sign for SORT-MERGE/3000. SORT-MERGE/3000 creates the output file, AMERICAN, of the correct size and type after the ›OUTPUT AMERICAN command is given. The command, ›KEY 31, 14, specifies a key, which starts in the position 31 and is fourteen characters long (records are sorted by occupations). The command, ›END, signals the end of the subsystem commands and initiates the SORT operation. Note the user input is underlined to distinguish it from the computer output.

```

/TEXT A, UNNUMBERED; LIST ALL, UNNUMBERED
Wiener,      Norbert  cybernetician  born 1894
Rothstein,   Arnold   gangster       born 1882
Clift,       Montgomery actor          born 1920
Truman,      Harry    politician     born 1884
Chamberlain, Wilt     sportsman     born 1936
Horse,       Crazy    warrior        born 1848
Joplin,      Janis    singer         born 1943
Vanderbilt, Cornelius capitalist     born 1794
Chavez,     Cesar    labor leader   born 1927
Crane,      Hart     poet           born 1899
/EXIT

```

*listing of the
file A by
using EDIT/3000*

```

END OF SUBSYSTEM
:RUN SORT.PUB.SYS

```

tells MPE to run SORT.

```

HP32214C.02.02 SORT/3000 FRI, SEP 19, 1980, 3:42 PM
(C) HEWLETT-PACKARD CO. 1980

```

```

>INPUT A
>OUTPUT AMERICAN
>KEY 31, 14
>END

```

*specifies the input file, A.
names the file that receives the sorted records.
describes a key.
tells SORT-MERGE/3000 to proceed with SORT.*

STATISTICS

```

NUMBER OF RECORDS = 10
NUMBER OF INTERMEDIATE PASSES = 0
SPACE AVAILABLE (IN WORDS) = 11,090
NUMBER OF COMPARES = 34
NUMBER OF SCRATCHFILE IO'S = 8
CPU TIME (MINUTES) = .00
ELAPSED TIME (MINUTES) = .01

```

```

END OF PROGRAM
:EDITOR
HP32201A.7.04 EDIT/3000 FRI, SEP 21, 1979, 3:42 PM
(C) HEWLETT-PACKARD CO. 1978

```

```

/TEXT AMERICAN, UNNUMBERED; LIST ALL, UNNUMBERED
Clift,      Montgomery actor          born 1920
Vanderbilt, Cornelius  capitalist     born 1794
Wiener,     Norbert   cybernetician  born 1894
Rothstein,  Arnold    gangster       born 1882
Chavez,     Cesar     labor leader   born 1927
Crane,      Hart      poet           born 1899
Truman,     Harry     politician     born 1884
Joplin,     Janis     singer         born 1943
Chamberlain, Wilt     sportsman     born 1936
Horse,      Crazy     warrior        born 1848

```

*listing of
the file
AMERICAN*

Figure 1-2. Running the Stand-Alone SORT Program

In Fig. 1-3, the file, PEOPLE, is sorted by using three keys; the last names, the first names, and the telephone numbers. The first key (positions 21 through 40) consists of the last names, the second key (positions 1 through 20) consists of the first names, and the third key (positions 41 through 53) consists of the telephone numbers. The sorted records are stored in a file, PHONBOOK. The VERIFY command lists the various options in effect during the SORT operation.

```

:RUN SORT.PUB.SYS                tells MPE to run SORT.

HP32214C.02.02 SORT/3000 FRI, SEP 19, 1980, 4:57 PM
(C) HEWLETT-PACKARD CO. 1980

>INPUT PEOPLE                    names the file to be sorted.
>OUTPUT PHONBOOK                specifies the output file, PHONBOOK.
>KEY 21, 20; 1, 20; 41, 13     describes three keys.
>VERIFY                          instructs SORT-MERGE/3000 to display the result
                                 of the commands typed in so far.

INPUT FILE = PEOPLE
RECORD LENGTH = SAME AS THAT OF THE INPUT FILE
OUTPUT FILE = PHONBOOK
KEY POSITION   LENGTH   TYPE   ASC/DESC
    21        20      BYTE   ASC   (MAJOR KEY)
    1         20      BYTE   ASC
    41        13      BYTE   ASC
>END                                                the end of the commands

```

Figure 1-3. Sorting the File PEOPLE

USING MERGE INTERACTIVELY

You can merge different sorted files by giving the following command:

```
:RUN MERGE.PUB.SYS
```

Two sorted files, AMERICAN and REST, are merged into a single file, WORLD (Fig. 1-4). In this case, both AMERICAN and REST are the input files. The subsystem creates the file, WORLD. Note the keys specified in the sorting operation (Fig. 1-2) have the same relative position and data format as those in the merging operation (Fig. 1-4). The >END command starts the merging operation.

```
:RUN MERGE.PUB.SYS
```

tells MPE to run MERGE.

```
HP32214C.02.02 MERGE/3000 FRI, SEP 19, 1980, 5:10 PM  
(C) HEWLETT-PACKARD CO. 1980
```

```
>INPUT AMERICAN, REST
```

specifies the sorted files, AMERICAN and REST.

```
>OUTPUT WORLD
```

names the output file, WORLD.

```
>KEY 31, 14
```

describes a key.

```
>END
```

the end of the commands.

STATISTICS

```
NUMBER OF INPUT FILES = 2  
NUMBER OF RECORDS = 20  
SPACE AVAILABLE (IN WORDS) = 11,164  
NUMBER OF COMPARES = 18  
CPU TIME (MINUTES) = .00  
ELAPSED TIME (MINUTES) = .01
```

```
END OF PROGRAM
```

```
:EDITOR
```

```
HP32201A.7.04 EDIT/3000 FRI, SEP 21, 1979, 5:10 PM  
(C) HEWLETT-PACKARD CO. 1978
```

```
/T WORLD, UNN; L ALL, UNN
```

Clift,	Montgomery	actor	born 1920
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894
Nijinsky,	Vaslav	dancer	born 1890
Khan,	Jenghiz	emperor	born 1167 (?)
Rothstein,	Arnold	gangster	born 1882
Chavez,	Cesar	labor leader	born 1927
Noether,	Emmy	mathematician	born 1882
Sen,	Mrinal	movie director	born 1923
Lautreamont,	Comte de	novelist	born 1846
Hammarskjold,	Dag	pacifist	born 1905
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Crane,	Hart	poet	born 1899
Truman,	Harry	politician	born 1884
K'ung,	Ch'iu	preacher	born 551 B.C.
Joplin,	Janis	singer	born 1943
Djilas,	Milovan	sociologist	born 1911
Chamberlain,	Wilt	sportsman	born 1936
Horse,	Crazy	warrior	born 1848

listing of the

file WORLD

Figure 1-4. Merging the Files AMERICAN and REST

Fig. 1-5 shows the merging of the two sorted files, PHONBK1 and PHONBK2 into the file, NEWBOOK.

```
:RUN MERGE.PUB.SYS           tells MPE to run MERGE.

HP32214C.02.02 MERGE/3000 FRI, SEP 19, 1980, 5:24 PM
(C) HEWLETT-PACKARD CO. 1980

>INPUT PHONBK1, PHONBK2       specifies the input files, PHONBK1 and PHONBK2.
>OUTPUT NEWBOOK            names the file that receives the merged records.
>KEY 21, 20                  describes three keys
>KEY 1, 20
>KEY 41, 13
>VERIFY                       lists the result of the commands typed in so far.

INPUT FILES = PHONBK1,PHONBK2
OUTPUT FILE = NEWBOOK
KEY POSITION   LENGTH   TYPE   ASC/DESC
      21       20     BYTE   ASC   (MAJOR KEY)
      1        20     BYTE   ASC
      41       13     BYTE   ASC

>END                           tells SORT-MERGE/3000 to proceed with MERGE.
```

Figure 1-5. Running the Stand-Alone MERGE Program

The examples described in this section familiarize you with an input file, what happens when it is sorted into an output file, and how this sorted file merges with another similarly sorted file to form a single sequential file.

RUNNING SORT AND MERGE AS STAND-ALONE PROGRAMS

SECTION


II

The various commands that perform the sorting and merging operations on files are described in this section. In the previous section, you have been briefly exposed to the simpler aspects of some of these commands—namely, the INPUT, OUTPUT, KEY, and END commands. The use of all the available options of these and other SORT-MERGE/3000 commands is explained in alphabetic order. The format of these commands, except the INPUT and OUTPUT commands, is identical for both SORT and MERGE.

The SORT and MERGE programs can be run during an interactive session or in a batch job. In an interactive session, they display the prompt character > during their execution and the commands are then typed in from the terminal. When large amounts of input and output are involved, it may be more convenient to run the program as a separate job; for example, streamed from a terminal.

When the length of a command exceeds one record, you may enter an ampersand (&) as the last non-blank character of the record and continue the command onto the next record. In an interactive session, SORT and MERGE prompt for the rest of the command with the >> continuation prompt.

The following is a list of the SORT-MERGE/3000 commands:

COMMAND	SYNTAX	DESCRIPTION
ALTSEQ	<p>>A[LTSEQ] <i>modspec1</i> [, <i>modspec2</i>]... [, <i>modspecN</i>]</p> <p>where <i>modspec</i> has the form:</p> $[\text{EACH}] \textit{leftspec} \left\{ \begin{array}{c} = \\ \text{WITH} \end{array} \right\} \textit{rightspec}$ <p style="text-align: center;">, or</p> $\text{MERGE} \textit{leftspec} \left\{ \begin{array}{c} \text{WITH} \\ = \end{array} \right\} \textit{rightspec}$ <p>and <i>leftspec</i> and <i>rightspec</i> have the form:</p> $\left\{ \begin{array}{l} \textit{string} \\ \textit{num byte} \\ \textit{range string} \end{array} \right\}$	<p>Allows you to define a collating sequence by modifying the basic ASCII (or EBCDIC) collating sequence.</p> 
DATA	<p>>DATA [IS] $\left\{ \begin{array}{l} \text{A[SCII]} \\ \text{E[BCDIC]} \end{array} \right\} [.] \text{SEQ[UENCE]}$</p> <p>[IS] $\left\{ \begin{array}{l} \text{A[SCII]} \\ \text{E[BCDIC]} \end{array} \right\}$</p>	<p>Specifies the type (ASCII or EBCDIC) of the input data and the basic collating sequence used in the sorting (or merging) operation.</p>
END	>E[ND]	<p>Concludes the specification of SORT (or MERGE) parameters and starts the operation.</p>

COMMAND	SYNTAX	DESCRIPTION
EXIT	>EX[IT]	Allows you to exit from SORT-MERGE/3000 and prevents any SORT (or MERGE) from being performed.
INPUT (sort)	>I[NPUT] $\left\{ \begin{array}{l} * \\ \$STDIN[X] \\ fname \\ (fname1, fname2, \dots, fnameN) \end{array} \right\}$ [, #records] [, rec size]	Specifies the input file(s) to be sorted.
INPUT (merge)	>I[NPUT] $\left\{ \begin{array}{l} \$STDIN[X] \\ filename1, filename2 \\ [filename3] \dots [filenameN] \end{array} \right\}$	Specifies the sorted files to be merged.
KEY	>K[EY] <i>keyspec 1</i> [, <i>keyspec 2</i>]...[, <i>keyspec N</i>] where <i>keyspec</i> has the form: <i>position, length</i> [, <i>type</i>][, DESC]	Defines the location of keys in the records
LANGUAGE	>L[ANGUAGE][IS] $\left\{ \begin{array}{l} langnum \\ langname \end{array} \right\}$	Defines the native language whose collating sequence is to be used to sort keys of type CHARACTER.
OUTPUT (sort)	>O[UTPUT] $\left\{ \begin{array}{l} filename \\ * \\ \$STDLIST \end{array} \right\}$ [, NUM][, KEY]	Creates the output file which receives the sorted records.
OUTPUT (merge)	>O[UTPUT] $\left\{ \begin{array}{l} filename [, num records][, KEY] \\ \$STDLIST \end{array} \right\}$	Creates the output file which receives the merged records.
RESET	>RESET	Allows you to correct errors in the KEY command(s).
SHOW	>SH[OW] $\left\{ \begin{array}{l} S[EQUENCE][, O[FFLINE]] \\ T[ABLE][, O[FFLINE]] \\ NOS[EQUENCE] \\ NOT[ABLE] \end{array} \right\}$	Displays the collating sequence or the translation table.
VERIFY	>V[ERIFY]	Lists the various options in effect during the particular SORT (or MERGE) operation.
	>: [MPE command]	Used to enter system commands from within SORT-MERGE/3000.

FILE DEFINITIONS

The SORT and MERGE programs reference some or all of the following files:

Display file	Receives the output (translation table or collating sequence) from the SHOW command. The formal designator of the display file is DISPLOUT which defaults to \$STDLIST.
Input file	Contains the records to be sorted or merged. It can be any file capable of sequential storage such as a file on magnetic tape, disc, or punched cards. The formal designator of the input file is INPUT, which is equated to the actual file designator you specify with the INPUT command. The input file is opened with the multirecord access option (aoption) which may be overridden with a file equation. (Multirecord access option is not supported on the HP 3000 Series I computer.) \$NULL is not a valid input file.
List file	Used by SORT-MERGE/3000 to send information (such as error messages) to you and to prompt for input in an interactive session. You should not confuse the list file with the output file which contains sorted (or merged) records. LIST is the formal file designator of the list file which defaults to \$STDLIST.
Output file	Received the sorted or merged records. An output file can consist of all the records of the input file(s) or only the key fields of the records. Its formal designator is OUTPUT, which is equated to the actual file designator you specify with the OUTPUT command. The output file is opened with the multirecord access option which may be overridden with a file equation. (Multirecord access option is not supported on the HP 3000 Series I Computer.)
Scratch file	The SORT program needs this disc file (named SORTSCR) to do the sorting. It is important to know this in case of errors. See the discussion under INPUT (sort) to estimate the scratch file size. All extents for the file are allocated at once. The MERGE program does not use this file.
Text file	Used to read SORT-MERGE/3000 commands directly from the file. TEXT is the formal file designator of the text file which defaults to \$STDINX.
Prompt file	Used by SORT-MERGE/3000 to prompt the user for input when the TEXT file is the session terminal but the LIST file is not. PROMPT is the formal file designator which defaults to \$STDLIST. The PROMPT file is the session output device.

ALTSEQ

Allows you to define a collating sequence by modifying the basic ASCII (or EBCDIC) collating sequence. It is effective only if the keys are of *type* BYTE, if the input data is ASCII, and if the DATA command has been previously issued.

SYNTAX

>A[LTSEQ] *modspec1* [, *modspec2*]...[, *modspecN*]

PARAMETERS

modspec A group of parameters used to define your own special collating sequence. You can successively use more than one such group of parameters in one or more ALTSEQ commands until the desired sequence is achieved.

where *modspec* has the following form:

$$\begin{array}{c} \text{[EACH] } \mathit{leftspec} \left\{ \begin{array}{c} = \\ \text{WITH} \end{array} \right\} \mathit{rightspec} \\ \text{or} \\ \text{MERGE } \mathit{leftspec} \left\{ \begin{array}{c} \text{WITH} \\ = \end{array} \right\} \mathit{rightspec} \end{array}$$

and *leftspec* and *rightspec* have the form:

$$\left\{ \begin{array}{l} \mathit{string} \\ \mathit{num byte} \\ \mathit{range string} \end{array} \right\}$$

EACH Indicates the collating sequence is to be modified by assigning each character of *leftspec* the ordinal value obtained by taking the ASCII code decimal value of the corresponding character in *rightspec*. If *leftspec* is longer than *rightspec*, *rightspec* is concatenated to itself enough times to make it equal in length to *leftspec*.

MERGE Indicates that the collating sequence is to be modified by merging *leftspec* and *rightspec*. Characters are selected alternatively from *leftspec* and *rightspec*.

If neither **EACH** nor **MERGE** is specified, the modification of the collating sequence is the same as if **EACH** has been specified except that *rightspec* is padded with blanks if it is shorter than *leftspec*.

= Functions as a separator between *leftspec* and *rightspec*.

WITH Can be used interchangeably with **=** and is generally used when **MERGE** is specified.

string A string of ASCII (or EBCDIC) characters enclosed in quotes. For example, "DAW".

num byte

A specification of the following form:

`[%[(bb)]] nnn`

bb is the base, which can be any decimal number between 2 and 16, inclusive. `%(bb)` must be specified in order to indicate a base other than 8 or 10. `%` indicates base 8 when no (*bb*) is specified. If both `%` and (*bb*) are omitted, the *nnn* parameter is assumed to be a decimal number (that is, base 10).

nnn represents an integer whose value is between 0 and decimal 255, inclusive. Each *n* is a digit between 0 and 9, inclusive, or one of the letters A, B, C, D, E, and F. The letters A through F are used to represent the digits 10 through 15, when a base greater than 10 is used. Each digit *n* of *nnn* must be less than the base *bb*.

For example, 12 represents the decimal value 12. `%12` represents the octal value 12, which is equivalent to the decimal value 10. `%(16)12` represents the hexadecimal value 12, which is equivalent to the decimal value 18.

range string

A specification of two characters separated by a minus sign and enclosed in quotes, or two numeric byte specifications separated by a minus sign. For example, "A-Z" or `%101-%132` (which specifies the same range as "A-Z").

Note whenever a minus sign is the second character in a group of three characters, the group is treated as a range. In all other cases, the minus sign is treated the same as any other character. For example, "A-D" represents the four characters A B C D while "AD-" represents the three characters A D -.

DISCUSSION

In each modification of the collating sequence, the ordinal values in the translation table assigned to the characters specified by *leftspec* are modified. (See the SHOW command for a discussion of the translation table.) If *rightspec* is longer than *leftspec*, the extra characters are ignored. If *leftspec* is longer than *rightspec* and neither EACH nor MERGE has been specified, *rightspec* is padded with blanks to make it equal in length to *leftspec*. For example, the command, `ALTSEQ "SAW"="TG"`, gives S, A, and W, the ordinal values T, G, and space. (See the discussions below for explanations of *modspec* with EACH and MERGE.) These assignments of new ordinal values are only for collating purposes. That is, the identity of the character is not lost; data is unchanged and appears in its original form in the output.

The DATA command specifying ASCII data and an ASCII or EBCDIC collating sequence must be issued before issuing the first ALTSEQ command in any SORT or MERGE operation. The error message, THE DATA COMMAND MUST BE ISSUED BEFORE THE ALTSEQ COMMAND CAN BE ISSUED, is displayed if the first ALTSEQ command is not preceded by a DATA command.

Note the operation of SORT (or MERGE) is somewhat slower when you have defined your own collating sequence by using the ALTSEQ command compared to the case when a standard ASCII or EBCDIC collating sequence is used.

modspec with EACH:

If EACH is specified, the modifications of the collating sequence are the same as explained above —except that, if *leftspec* is longer than *rightspec*, *rightspec* is concatenated to itself a sufficient number of times to make it equal in length to *leftspec*. For example, the command, `>ALTSEQ EACH "ADW"="FG"`, gives A, D, and W the ordinal values obtained by taking the ASCII code decimal values of F, G, and F. Assuming the basic collating sequence has been specified as ASCII, this means in the sixth row of the fifth column of the translation table will now appear A=70, in the sixth row of the eighth column will appear D=71, and in the eighth row of the seventh column will appear W=70. Note 70 and 71 are the ASCII code decimal values of the characters F and G, respectively. See Figs. 2-1 through 2-5 for more examples.

modspec with MERGE:

When MERGE is specified in the *modspec* parameter, the values in the translation table assigned to the characters specified by *leftspec* and *rightspec*, and the characters in between are modified. Characters are selected alternatively from *leftspec* and *rightspec* and the translation table is modified so the characters collate in this order. The first character is always selected from *leftspec*. If *leftspec* precedes *rightspec* in the collating sequence, the sequence is modified so the characters between the two ranges collate after the merger of the ranges. If *rightspec* precedes *leftspec*, the characters between the two specifications collate before the first character of the first range. When either range is exhausted, the characters from the other range are simply appended until that range is exhausted too. Note the strings specified by *leftspec* and *rightspec* must be strictly increasing and contiguous whenever MERGE is specified.

If you wish to do an alphabetic sorting in which each upper case letter collates ahead of the corresponding lower case letter, use the command, `>ALTSEQ MERGE "A-Z" WITH "a-z"`. The following six special characters follow the lower case z since the first range precedes the second range:

[\] ^ _ `

If the *modspec* is, MERGE "a-z" WITH "A-Z", the same six characters precede the lowercase a. Refer to Figs. 2-6 through 2-9 for more examples.

You may consider this form of *modspec* as a shorthand for the *modspec* specifying EACH. For example, the command, `>ALTSEQ MERGE "A-Z" WITH "a-z"`, is equivalent to the lengthier command, `>ALTSEQ "AaBb...Zz"="AB...Zab...z"`, where...represent all the necessary characters.

DATA

Specifies the type (ASCII or EBCDIC) of the input data and the basic collating sequence used in the particular SORT (or MERGE) operation. The collating sequence may or may not be altered further by the ALTSEQ command.

SYNTAX

```
>DATA [IS] { A[SCII]
              E[BCDIC] } [,] SEQ[UENCE] [IS] { A[SCII]
                                              E[BCDIC] }
```

DISCUSSION

This command must precede the first ALTSEQ command in any SORT or MERGE operation because the DATA command always initializes the translation table. The message, THE DATA COMMAND MUST BE ISSUED BEFORE THE ALTSEQ OR SHOW COMMANDS is displayed, if the first ALTSEQ command is not preceded by the DATA command. If the DATA command is reissued, following an ALTSEQ command, the translation table (and the collating sequence) are reset to their original status.

The specification of a particular sequence is only for collating purposes. A user defined sequence can be designated only if the input data is ASCII. The input data is unchanged and appears in the output in its original form. The following example shows how the DATA command nullifies the effect of the ALTSEQ command issued previously during a SORT operation

EXAMPLE

```
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SAT, SEP 20, 1980, 9:54 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>ALTSEQ MERGE "A-T" WITH "V-Y"
```

```
THE DATA COMMAND MUST BE ISSUED BEFORE THE ALTSEQ OR SHOW COMMANDS CAN BE ISSUED.
```

```
>DATA IS ASCII, SEQUENCE IS ASCII
```

```
>ALTSEQ MERGE "A-T" WITH "V-Y"
```

```
>SHOW SEQUENCE
```

nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	lf	vt	ff	cr	so	si
dle	dc1	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
sp	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	V	B	W	C	X	D	Y	E	F	G	H	I	J	K
L	M	N	O	P	Q	R	S	T	U	Z	[\]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	del

>DATA IS ASCII, SEQUENCE IS ASCII

>SHOW SEQUENCE

nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	lf	vt	ff	cr	so	si
dle	dc1	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
sp	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	del

END

Concludes the specification of SORT (or MERGE) parameters and starts the operation.

SYNTAX

>E[ND]

DISCUSSION

The END command indicates there are no more commands and the SORT (or MERGE) program should begin. If * (or \$STDIN) is specified in the INPUT command of the SORT program, the character, ?, is displayed following the END command, and the input records are typed in from the terminal.

EXIT

Allows you to exit from SORT-MERGE/3000 and prevents any SORT (or MERGE) from being performed.

SYNTAX

>EX[IT]

EXAMPLE

```
>INPUT A
>OUTPUT NEW
>KEY 1, 15
>EXIT

END OF PROGRAM
:EDITOR
HP32201A.7.04 EDIT/3000 SAT, SEP 22, 1979, 10:10 PM
(C) HEWLETT-PACKARD CO. 1978
/T NEW, UNN

+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! ERROR NUMBER: 52 RESIDUE: 0 !
! BLOCK NUMBER: 0 NUMREC: 0 !
+-----+
*23*FAILURE TO OPEN TEXT FILE (52)
```

Note the output file, NEW, is not created, as the EXIT comand disallows the sorting of the input file, A.

INPUT

(sort)

Specifies the input file(s) to be sorted.



SYNTAX

$$>I[INPUT] \left\{ \begin{array}{l} * \\ \$STDIN[X] \\ \mathit{fname} \\ (\mathit{fname1}, \mathit{fname2}, \dots, \mathit{fnameN}) \end{array} \right\} [, \#records] [, \mathit{rec\ size}]$$

PARAMETERS

*(or \$STDIN[X]) Specifies that the input records are read from the text file TEXT (or a job standard input device — that is, a card reader, tape, magnetic disc for a streamed job, or terminal) instead of the input file.

fname Actual file designator. \$Null is not a valid input file.

#records A positive integer specifying the upper limit of the number of records sorted. It is the sum of the number of records of each input file, if multiple input files are specified. *#records* should be specified only if one or more input files are not disc files. When the input file is a disc file, its current end-of-file (EOF) value is used. *#records* is ignored in this case. If the input file is not a disc file and the *#records* is not specified, a default value of 10,000 is assumed by SORT. This parameter cannot be used to extract a subset of the input file. You may use the FCOPY utility to accomplish this.

rec size A positive integer specifying the number of maximum allowable characters in a record. This is important only if the records are of variable lengths. Additionally, if the scratch file record size is limited, *rec size* should be set equal to the size of the largest record.

If you want to estimate the scratch file record size (SFRS) and the scratch file size (SFS), use the following equations:

$$SFRS = ((rec\ size + 7)/2) + 4$$

where *rec size* is the input record size in bytes. (You must add the length of the keys to the *rec size* if the keys are of the type, BYTE, and ALTSEQ is used.) SFRS is in words.

$$SFS = ((SFRS * \#records)/128) + 1$$

SFS is in sectors.

You can issue a file equation for the scratch file only to specify a particular logical device which must be a disc. For example, FILE SORTSCR; DEV=2

DISCUSSION

During SORT, the INPUT command specifies one or more files containing the records to be sorted. In the case of multiple input files, all the files are sorted and merged into a single output file. If * (or \$STDIN) is specified, input records are assumed to follow the END command and continue until EOF is reached (indicated by typing :EOD). If input records are entered from the terminal, the prompt character ? is displayed for each record. Note when you specify more than one input file during SORT, the names of all the input files must be enclosed in parentheses. This differs from the use of the INPUT command during MERGE, where parentheses cannot be used. If more than one INPUT command is entered, only the last command is effective. Thus, all the files to be sorted must be specified in a single INPUT command. This command can be entered any time before the END command. In the absence of the INPUT command, any permanent disc file with the formal designator INPUT is considered the input file. Also, SORT-MERGE/3000 does not disallow file equations issued prior to your entering the subsystem. So, if the INPUT command refers to the same file as specified in the file equation, the file's characteristics are determined by the file equation. You should give the :RESET command before entering SORT-MERGE/3000 if you want the default values for the parameters of the file. The same holds for the OUTPUT command during SORT and the INPUT and OUTPUT commands during MERGE.

EXAMPLE

```
>INPUT R,, 30
```

The file, R, is to be sorted with a maximum of 30 characters from each record.

INPUT

(merge)

Specifies the sorted files to be merged.


SYNTAX

```
>I[INPUT] { $STDIN  
           filename1, filename2 } [, filename3]...[, filenameN]
```

PARAMETERS

<i>\$STDIN</i>	Specifies that the records of the sorted input files are entered from the terminal in a session (or a job standard input device in a batch mode).
<i>filename</i>	Actual file designator. \$Null is not a valid input file.

DISCUSSION

Unlike the INPUT command during SORT, the input files cannot be enclosed in parentheses in this case. The order in which the files are specified is relevant only in that the records with equal keys are ordered according to the order of the files in which they appear. If more than one INPUT command is entered, only the last command is effective. It may be entered any time before the END command. Note, \$STDIN may also be used as an input file. In this case the records are assumed to follow the END command and must have the same length. "?" is not displayed (cursor keeps blinking if you are using a CRT terminal). Input is terminated by :EOD. Merge does not allow the use of "*" 

EXAMPLE

```
>INPUT A, B, C
```

A, B, and C are the three files to be merged.

KEY

Defines the location of keys in the records.

SYNTAX

>K[EY] *keyspec1* [; *keyspec2*][; *keyspec3*]...[; *keyspecN*]

PARAMETERS

keyspec A group of parameters used to specify the keys

SYNTAX *position, length* [, *type*][, DESC]

position A positive integer specifying the position of the first character of the key field.
(The first position of the record is numbered one.)

length A positive integer indicating the length of the key field in bytes.

type Defines the type of data contained in the key fields and can be one of the following mnemonics:

B[YTE]

Direct byte comparison is used. It is the default value for the *type* parameter and should be used for ASCII, EBCDIC, or logical quantities.

I[NT]

Key field contains a 2's complement number of the specified length in bytes. *length* defaults to two bytes. Any value may be specified for *length*.

DO[UBLE]

Same as the INT mnemonic but *length* defaults to four bytes.

R[EAL]

Key field contains a floating point number. Any value may be specified for *length*. *length* defaults to four bytes.

L[ONG]

Same as REAL. Any value may be specified for *length*. *length* defaults to eight (or six, if your system is supervised by MPE-C) bytes.

P[ACKED]

Key field contains a packed decimal number. In this format, each character except the last, contains two digits. Each digit occupies four bits. The last character contains the least significant digit of the number in its four leftmost bits, and the sign of the number in its four rightmost bits. The sign is considered minus if it has the value 1101, and plus otherwise.

PACKED*

Same as PACKED except there are only an even number of digits and a sign. The four higher ordered digits are not treated as a part of the field.

DI[SPLAY-TRAILING-SIGN]

Key field contains a numeric display quantity. Numeric display items are represented by ASCII coded decimal digits (0 through 9) except for the units digit which carries the sign of the data item. The sign is determined according to the Table 2-1. For example, 123 is represented by 12C. (This is the same as DISPLAY in the previous versions of SORT-MERGE/3000.)

DISPLAY-L[EADING SIGN]

In this case, the first digit carries the sign of the data item. For example, -123 is represented by J23.

DISPLAY-TRAILING-SIGN-S[EPARATE]

The sign is contained in the character position to the right of the units digit. For example, 123 is represented by 123+.

DISPLAY-LEADING-SIGN-S[EPARATE]

The sign is contained in the character position to the left of the first digit. For example, -123 is represented by -123.

DESC

Indicates the records are arranged in a descending order. If this parameter is not specified, the records are arranged in the ascending order.

C[HARACTER]

The collating sequence for the native language defined in the LANGUAGE command is to be used.

Display Digit	Positive	Negative	No Sign
0	{ (%173)	} (%175)	0 (%60)
1	A (%101)	J (%112)	1 (%61)
2	B (%102)	K (%113)	2 (%62)
3	C (%103)	L (%114)	3 (%63)
4	D (%104)	M (%115)	4 (%64)
5	E (%105)	N (%116)	5 (%65)
6	F (%106)	O (%117)	6 (%66)
7	G (%107)	P (%120)	7 (%67)
8	H (%110)	Q (%121)	8 (%70)
9	I (%111)	R (%122)	9 (%71)

Table 2-1. Internal Representation in ASCII

DISCUSSION

SORT-MERGE/3000 sorts keys containing Binary, ASCII, or EBCDIC data according to an eight-bit binary sequence (00000000 to 11111111), except for the type **CHARACTER**, which is sorted according to the collating sequence for the native language specified in the **LANGUAGE** command. See Appendix C for further information on native language collating. Other types of data (integer, real, etc.) are sorted according to the standard arithmetic relational operators. For example, 2 is greater than -5. The keys can contain alphabetic, numeric, or alphanumeric (alphabetic and numeric intermixed) data. They can be contiguous or separated in a record or they can overlap each other; provided the collating sequence is not altered, or a user defined sequence is not used. An entire record can be considered as a single key.

As explained in Section I, each **KEY** command can specify one or more key fields and the specifications are separated by semicolons. Multiple key fields can also be specified with more than one **KEY** command. All the key fields do not have to be specified in the same command. The most significant key is called the major key and is declared first in the command. Other keys have decreasing significance according to their relative positions following the major key. They are compared if a comparison of more significant keys results in an equal condition.

Consider a file containing the records of all the students in a high school. Each record can contain such information as name, address, grade level, grades in individual courses etc. You can specify the order in which the records are sorted. If the first record is of the student with highest grades (A) in English and Math, you specify an ascending order. If the major key is English and the other key is Math, the data in the Math fields are compared only if the data in the English fields are the same. The sorting order is specified in the same commands that specify the keys. An order is declared for each key. This order does not have to be the same for all the keys in a record. For example, in the high school file, you can declare English (major key) with an ascending order and Math with a descending order. Note even if the sorting order is different for each key, only one collating sequence is used for a particular operation.

EXAMPLES

- | | |
|----------------------------------|---|
| >KEY 10, 5 | BYTE key of length 5 starting in position 10, sorted in the ascending order |
| >KEY 20, REAL | REAL key of length 4, starting in position 20 and sorted in an ascending order since four is the default for the <i>length</i> parameter when the key data type is REAL |
| >KEY 30, 20, INT, DESC | 20-byte INTEGER key starting in position 30, and sorted in a descending order |

(See the **RESET** command to make corrections to the keys.)

LANGUAGE |

Defines the native language whose collating sequence is to be used to sort keys of type CHARACTER.

SYNTAX

```
> L[LANGUAGE][IS] { langnum  
                   }  
                   { langname }
```

PARAMETERS

langnum A language id number. This must specify a language configured on the system

langname A language name. This must specify a language configured on the system

DISCUSSION

The LANGUAGE command causes SORT-MERGE to sort keys of type CHARACTER according to the collating sequence of the language specified by the *langnum* or *langname*. The Native Language Support intrinsics and files must first be installed on the system before the LANGUAGE command can be used. See Appendix C in this manual and the Native Language Support Reference Manual for more information.

The LANGUAGE command does not affect SORT-MERGE messages, syntax or prompts.

EXAMPLES

```
LANGUAGE IS SPANISH  
LANG 5  
L FRENCH
```


OUTPUT

(sort)

Creates the output file which receives the sorted records.

SYNTAX

>O[UTPUT] $\left\{ \begin{array}{l} \textit{filename} \\ * \\ \$\text{STDLIST} \end{array} \right\} [, \text{NUM}][, \text{KEY}]$

PARAMETERS

filename Actual file designator

*(or \$STDLIST) Specifies that the sorted records are sent to the terminal during a session and terminal or line printer during a batch mode. Output file is not saved in this case.

NOTE: Use * for CCTL (carriage control, which causes the first byte to be stripped from each record), and use \$STDLIST for NOCCTL in interactive mode. In batch mode, both default to CCTL. The user may specify the following file equation to force NOCCTL:

:FILE LIST; DEV=LP; NOCCTL

NUM Specifies that the output records consist of the original logical record numbers. These are double word binary numbers which cannot be meaningfully printed or displayed on the terminal. This parameter must not be specified if you specify * (or \$STDLIST) in the command. The first record in the file is considered number one.

KEY Specifies that the output records consist of only the key fields.

If neither NUM nor KEY is specified, the output records are identical to the input records. If NUM is specified but KEY is not specified, the output records consist of a double integer whose value is the original logical (relative) record number. If KEY is specified and NUM is not specified, the output records consist of the key fields concatenated together from left to right. If both NUM and KEY are specified, the output records consist of the original logical record number and the key fields; concatenated together with the logical record number on the right.

EXAMPLE

OUTPUT REST, NUM

DISCUSSION

In the absence of this command, SORT-MERGE/3000 creates the output file by the file name, OUTPUT. In the event of several OUTPUT commands, only the last OUTPUT command is effective.

Sends the logical record numbers to the file, REST.

Note if a file already exists with the same name specified in the OUTPUT command, the following message is displayed if you are in an interactive session:

PURGE OLD OUTPUT FILE *filename*?

If your response is YES, the old file is purged. If this is not possible (for example, you cannot purge a file which is used by some other user at the same time), if your response is NO, or if you press *return*, the following message is displayed:

ENTER NEW NAME FOR OUTPUT FILE

In this case, you should enter a new name for the output file.

In a batch mode, the old file is not disturbed. Instead, a new permanent file OUTPUT nn (n is a non-negative integer) is created with the following message:

OUTPUT FILE CLOSED WITH FILE NAME OUTPUT nn

The system Job Control Word (JCW) is set to FATAL when an alternate output file is used.

OUTPUT

(merge)

Creates the output file which receives the merged records.

SYNTAX

>O[OUTPUT] { *filename* } [,*num records*] [,KEY]
 { \$STDLIST }

filename Actual file designator

\$STDLIST Records of the merged files are sent to the terminal during a session and line printer or terminal during a batch job. The output file is not saved in this case. In interactive mode the default is no carriage control(NOCCTL). In batch mode, the default is CCTL (first byte of each record is stripped). To force NOCCTL in batch mode issue the following file equation:

:FILE OUTPUT; DEV=LP; NOCCTL

and do not issue an OUTPUT command to MERGE.

num records A positive integer specifying the upper limit of the number of records merged. This should be large enough to include all the input files. If *num records* is specified, and if any one of the input files is not a disc file, this value is used as the *filesize* parameter during the opening of the output file. If one or more input files are not disc files and if *num records* is not specified, a default value of 10000 records is used by MERGE. This parameter is ignored if all the input files are disc files.

KEY Output consists of the key fields only.

EXAMPLES

OUTPUT FILE1,
KEY Sends only the keyfields of the merged records to the file, FILE1.

OUTPUT FILE2,
50000 Unless all the input files are disc files, the *filesize* parameter of the output file, FILE2, is taken as 50000, when the file is opened.

During the MERGE operation, the OUTPUT command specifies the file to which the merged records are written. If more than one OUTPUT command is entered, only the last command is effective.

If a file already exists with the same name as specified in the OUTPUT command, the following message is displayed if you are in an interactive session:

PURGE OLD OUTPUT FILE *filenme*?

If your response is YES, the old file is purged. If this is not possible (for example, you cannot purge a file which is used by some other user at the same time), if your response is NO, or if you press *return*, you get the following message:

ENTER NEW NAME FOR OUTPUT FILE

Like the OUTPUT command during a SORT, you should enter a new name for the output file.

In a batch mode, the old file is not disturbed. Instead, a new permanent file OUTPUT nn (n is a non-negative integer) is created with the following message:

OUTPUT FILE CLOSED WITH FILE NAME OUTPUT nn

RESET

Nullifies the existing **KEY** command(s). This command is used to correct errors in the key specification(s).

SYNTAX

>**RESET**

SHOW

Displays the collating sequence or the translation table.

SYNTAX

$$>SH[OW] \left\{ \begin{array}{l} S[SEQUENCE][, O[FFLINE]] \\ T[ABLE][, O[FFLINE]] \\ NOS[SEQUENCE] \\ NOT[ABLE] \end{array} \right\}$$

PARAMETERS

S[SEQUENCE] Displays the collating sequence.

This sequence is determined by the first 128 characters of the ASCII code, unless preceded by an ALTSEQ command or a DATA command with the EBCDIC sequence parameter. Without the OFFLINE parameter, the sequence is displayed on the terminal. (It is printed on the line printer, if the OFFLINE parameter is used.) The display consists of the representation of each character in the relative order in which the collating sequence sorts (or merges) the records. Characters with the same ordinal values are adjoined by equal sign(s). Once specified in the SHOW command, it is displayed after each subsequent ALTSEQ command during a particular SORT (or MERGE) operation until you specify NOSEQUENCE. OFFLINE activates the formal file designator DISPLOUT, with the line printer as the default device type (DEV=LP). Alternatively, you can store the contents of the sequence on a disc (or tape) file by appending DEV=DISC (or TAPE) to the file equation.

T[ABLE] Displays the translation table.

After defining your special collating sequence, you may want to look at the table and the changes that occur in it. The table is helpful if you call SORT (or MERGE) from a program (Sections III-VI). The translation table is organized according to the ASCII code decimal values of the characters. You should look at the position defined by the ASCII code decimal value to determine the ordinal value of a particular character. The table displays graphic characters, each equated to its ordinal value, and the ordinal values of the characters that do not have graphic representation. Like the SEQUENCE option, the translation table is displayed after each ALTSEQ command. The >SHOW TABLE command displays the table (in decimal) on the terminal.

NOS[SEQUENCE] Suppresses the display of the collating sequence in a particular SORT (or MERGE) operation. However, you can again get the display by specifying SEQUENCE.

NOT[ABLE] Suppresses the display of the translation table until you give the SHOW TABLE command.

Example of the SHOW command with the TABLE parameter

```
>DATA A SEQ A
>A "B" = "A"
>SHOW TABLE
```

TABLE OF ORDINAL VALUE ASSIGNED TO EACH CHARACTER.

!	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
0 !	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
1 !	10 !	11 !	12 !	13 !	14 !	15 !	16 !	17 !	18 !	19 !
2 !	20 !	21 !	22 !	23 !	24 !	25 !	26 !	27 !	28 !	29 !
3 !	30 !	31 !	sp=32	!!= 33	!="= 34	!#= 35	!\$= 36	!%= 37	!&= 38	!'= 39
4 !	(= 40)= 41	!= 42	!+= 43	!,= 44	!--= 45	!.= 46	!/= 47	!0= 48	!1= 49
5 !	2= 50	!3= 51	!4= 52	!5= 53	!6= 54	!7= 55	!8= 56	!9= 57	!:= 58	!;= 59
6 !	!<= 60	!--= 61	!>= 62	!?= 63	!@= 64	!A= 65	!B= 66	!C= 67	!D= 68	!E= 69
7 !	!F= 70	!G= 71	!H= 72	!I= 73	!J= 74	!K= 75	!L= 76	!M= 77	!N= 78	!O= 79
8 !	!P= 80	!Q= 81	!R= 82	!S= 83	!T= 84	!U= 85	!V= 86	!W= 87	!X= 88	!Y= 89
9 !	!Z= 90	![= 91	!\= 92	!] = 93	!^= 94	!_ = 95	!`= 96	!a= 97	!b= 98	!c= 99
10 !	!d=100	!e=101	!f=102	!g=103	!h=104	!i=105	!j=106	!k=107	!l=108	!m=109
11 !	!n=110	!o=111	!p=112	!q=113	!r=114	!s=115	!t=116	!u=117	!v=118	!w=119
12 !	!x=120	!y=121	!z=122	!{=123	! =124	!}=125	!~=126	! =127	! 128	! 129
13 !	130 !	131 !	132 !	133 !	134 !	135 !	136 !	137 !	138 !	139 !
14 !	140 !	141 !	142 !	143 !	144 !	145 !	146 !	147 !	148 !	149 !
15 !	150 !	151 !	152 !	153 !	154 !	155 !	156 !	157 !	158 !	159 !
16 !	160 !	161 !	162 !	163 !	164 !	165 !	166 !	167 !	168 !	169 !
17 !	170 !	171 !	172 !	173 !	174 !	175 !	176 !	177 !	178 !	179 !
18 !	180 !	181 !	182 !	183 !	184 !	185 !	186 !	187 !	188 !	189 !
19 !	190 !	191 !	192 !	193 !	194 !	195 !	196 !	197 !	198 !	199 !
20 !	200 !	201 !	202 !	203 !	204 !	205 !	206 !	207 !	208 !	209 !
21 !	210 !	211 !	212 !	213 !	214 !	215 !	216 !	217 !	218 !	219 !
22 !	220 !	221 !	222 !	223 !	224 !	225 !	226 !	227 !	228 !	229 !
23 !	230 !	231 !	232 !	233 !	234 !	235 !	236 !	237 !	238 !	239 !
24 !	240 !	241 !	242 !	243 !	244 !	245 !	246 !	247 !	248 !	249 !
25 !	250 !	251 !	252 !	253 !	254 !	255 !				

WHEN PASSED TO SORTINIT, THE TABLE ABOVE IS PRECEDED BY TWO BYTES. THESE FIRST TWO BYTES CONTAIN A FLAG BYTE OF X000 AND A LENGTH BYTE OF X377 RESPECTIVELY.

Columns are labeled 0, 1, 2, ..., 9, and rows are labeled 0, 1, 2, ..., 25. The table is used by first reading down the leftmost column and then across from left to right. If you want to know the current ordinal value of B (whose ASCII code decimal value is 66), read down the table to locate the row labeled 6. Then read across until you reach the column with the heading 6. The value (65) contained in this position (6, 6) identifies the location of the character B in the altered collating sequence.

You can use the OFFLINE parameter to send the contents of the table to the line printer, disc, or tape. In this case, the table is created in three forms. During the programmatic usage of SORT-MERGE/3000, this information is edited and inserted into a program and then copied into the altseq array passed to SORTINIT (or MERGEINIT).

See Figs. 2-11 through 2-17 for more examples.

VERIFY

Lists the various options in effect during the particular SORT (or MERGE) operation.

SYNTAX

>V[ERIFY]

EXAMPLES

SORT operation:

>VERIFY

```
INPUT FILE = MYTHO
RECORD LENGTH = SAME AS THAT OF THE INPUT FILE
OUTPUT FILE = FICTION
KEY POSITION   LENGTH   TYPE   ASC/DESC
      22       10     BYTE   ASC   (MAJOR KEY)
      1        3     BYTE   ASC
```

INPUT DATA IS IN ASCII.
SEQUENCE IS IN EBCDIC.

The file, MYTHO, is to be sorted into the file, FICTION, with EBCDIC as the collating sequence.

MERGE operation:

>VERIFY

```
INPUT FILES = AMERICAN,REST
OUTPUT FILE = WORLD
KEY POSITION   LENGTH   TYPE   ASC/DESC
      1       15     BYTE   ASC   (MAJOR KEY)
     31       14     BYTE   ASC
```

The files, AMERICAN and REST, are to be merged into the file, WORLD. Note the collating sequence is ASCII by default.



The : command is used to enter MPE commands from within SORT or MERGE.

SYNTAX

>: [MPE command]

The : command allows you to enter certain MPE commands without using the BREAK key. The colon indicates to SORT-MERGE/3000 that it should pass the rest of the record to the MPE operating system. To continue an MPE command on the next record, the last non-blank character on the current record should be an ampersand (&). The command may be continued after the >: prompt.

Valid MPE commands are those which can be executed programmatically (see the MPE INTRINSICS MANUAL, page 4-9 for a list of such commands). Command interpreter and file system error messages will be printed if an error occurs. User Defined Commands are not available from the : command, although they are valid during a BREAK.

EXAMPLE

***** MPE COMMAND EXAMPLE *****

:RUN SORT.PUB.SYS

HP32214C.02.03 SORT/3000 TUE, JAN 29, 1980, 11:06 AM

(C) HEWLETT-PACKARD CO. 1980

>:BUILD LPFILE;REC=-132,10,F,ASCII; &

DISC=10000,32,32;CCTL

>:LISTF LPFILE,2

TUE, JAN 29, 1980, 11:06 AM

ACCOUNT= SUBSYS GROUP= SORT

FILENAME	CODE	-----LOGICAL RECORD-----			----SPACE----		
		SIZE	TYP	EOF	LIMIT R/B	SECTORS	#X MX
LPFILE		133B	FAC	0	10000 10	6006	32 32

>EXIT

END OF PROGRAM

CONTROL Y

During the running of a SORT or MERGE program in an interactive session, you can obtain its status by typing Y^c. For example, the displayed status may be similar to one of the following messages:

INPUT PHASE: 1234 RECORDS HAVE BEEN INPUT

OUTPUT PHASE: 9 RECORDS HAVE BEEN OUTPUT.

OUTPUT PHASE: 0 RECORDS HAVE BEEN OUTPUT.

INTERMEDIATE SORT PHASE: PASS 1 OF 3. (675 RECORDS MERGED)

:EOD

Terminates your input records when terminal is the input device.



EXAMPLES

TYPES OF MODIFICATION SPECIFICATIONS

The ALTSEQ command with EACH followed by a *string spec*

```
:RUN SORT.PUB.SYS

HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 11:56 AM
(C) HEWLETT-PACKARD CO. 1980

>DATA IS ASCII, SEQUENCE IS ASCII
>ALTSEQ EACH "LMN"="ST"
>SHOW SEQUENCE
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R
L= N= S M= T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del
```

Figure 2-1

The result of *modspec* EACH "LMN"="ST"

<u>Original list</u>	<u>The list during SORT</u>	<u>Sorted result</u>
TOKEN	TOKES	COST
MOP	TOP	COME
COST	COST	SING
COME	COTE	NOSE
TABLE	TABSE	LONESOME
MISS	TISS	SOLE
SING	SISG	TABLE
NOSE	SOSE	MISS
LONESOME	SOSESOTE	TOKEN
SOLE	SOSE	MOP

Figure 2-2

During the SORT operation, L and N are equated to S, and M is equated to T.

The ALTSEQ command without using EACH

```

>DATA A SEQ A
>A "ABC" = "X"
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp= B= C ! " # $ % & ' ( ) * + , - . /
> ? @ D E F G H I J K L M N O P
Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del

```

Figure 2-3

The ALTSEQ command pads X with two blanks to make it equal to ABC in length. Note the character sp is equated to B and C and the character A to X, in the collating sequence. The table position identified by each character of the left string is replaced by the corresponding character of the right string until the string ABC is exhausted.

Numeric byte specification

```

>DATA A SEQ A
>A 65=%141
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ B C D E F G H I J K L M N O P
Q R S T U V W X Y Z [ \ ] ^ _ `
A= a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del

```

Figure 2-4

A is assigned the same ordinal value as a in the final collating sequence.

Range string specification

```
>A %101-%132="a-z"
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ [ \ ] ^ _ ` A= a B= b C= c D= d E=
e F= f G= g H= h I= i J= j K= k L= l M=
m N= n O= o P= p Q= q R= r S= s T= t U=
u V= v W= w X= x Y= y Z= z { | } ~ del
```

Figure 2-5

The left range is specified by two numeric byte specifications separated by a minus sign. Note the same range can be represented by "A-Z", %101-"Z", or 65-90.

Collating upper-lower case alphabetic characters

```
>A MERGE "A-Z" WITH "a-z"
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A a B b C c D d E e F f G g H h
I i J j K k L l M m N n O o P p
Q q R r S s T t U u V v W w X x
Y y Z z [ \ ] ^ _ ` { | } ~ del
```

Figure 2-6

The six characters [, \ ,] , ^ , _ , ` follow the lower case z, as the first range precedes the second range.

Collating lower-upper case alphabetic characters

```
>A MERGE "a-z" = "A-Z"
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ [ \ ] ^ _ ` a A b B c C d D e
E f F g G h H i I j J k K l L m
M n N o O p P q Q r R s S t T u
U v V w W x X y Y z Z { | } ~ del
```

Figure 2-7

The six characters [, \ ,] , ^ , _ , ` precede the lowercase a.

The result of MERGE "a-z" WITH "A-Z"

<u>Original list</u>	<u>Sorted list using MERGE</u>
CAN	AXE
shovel	boy
MAN	BROOM
BROOM	CAN
TABLE	drawer
AXE	DOG
drawer	MAN
boy	shovel
DOG	TABLE

Figure 2-8

Merging unequal strings

```
>A MERGE "ABCD" WITH "ab"
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A a B b C D E F G H I J K L M
N O P Q R S T U V W X Y Z [ \ ]
^ _ ` c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del
```

Figure 2-9

The collating sequence appears as AaBbCDEF...Z. The merging of the strings continue until the right string "ab" is exhausted.

Using the ALTSEQ command in a batch mode

```
:JOB USER.ACCT
JOB NUMBER = #J5
SUN, SEP 21, 1980, 12:21 PM
HP3000 / MPE III C.00.02
```

```
:EDITOR
HP32201A.7.04 EDIT/3000 SUN, SEP 23, 1979, 12:21 PM
(C) HEWLETT-PACKARD CO. 1978
T UNDRGRAD, UNN; L ALL, UNN
/T UNDRGRAD, UNN; L ALL, UNN
```

Virgin Cat	3.1	A
Tech Nitpicker	3.2	A
Sensible Kommunist	3.6	B
Boris Frankenstein	3.1	A
Milind Ranade	3.9	B
Uncle Samuelson	3.7	B
Thomas Collins	2.1	U
Vegetarian Dracula	3.8	B
Homo Genius	3.4	A
Hit Woman	3.1	A
Sorting Jack	3.3	A
Harry Krishna	2.9	U
Lacy Lowercase	3.4	A
Nicolas Bourbaki	4.0	B
Red Butler	3.1	A

E

/E

```
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 12:28 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
DATA A SEQ A
A "BA" = "AB"
INPUT UNDRGRAD
OUTPUT VICTORS
KEY 38, 1; 1, 3
END
```

STATISTICS

NUMBER OF RECORDS =	15
NUMBER OF INTERMEDIATE PASSES =	0
SPACE AVAILABLE (IN WORDS) =	10,958
NUMBER OF COMPARES =	60
NUMBER OF SCRATCHFILE IO'S =	10
CPU TIME (MINUTES) =	.00
ELAPSED TIME (MINUTES) =	.01
RECORD SIZE (IN BYTES) =	72
SCRATCH FILE SIZE (# SECTORS) =	83

```

:EDITOR
END OF PROGRAM
HP32201A.7.04 EDIT/3000  SUN, SEP 23, 1979, 12:30 PM
(C) HEWLETT-PACKARD CO. 1978
T VICTORS, UNN; L ALL, UNN
/T VICTORS, UNN; L ALL, UNN
Nicolas Bourbaki      4.0      B
Milind Ranade         3.9      B
Sensible Kommunist    3.6      B
Uncle Sammuelson     3.7      B
Vegetarian Dracula    3.8      B
Boris Frankenstein    3.1      A
Hit Woman             3.1      A
Homo Genius           3.4      A
Lacy Lowercase        3.4      A
Red Butler            3.1      A
Sorting Jack          3.3      A
Tech Nitpicker        3.2      A
Virgin Cat            3.1      A
Harry Krishna         2.9      U
Thomas Collins        2.1      U
:EOJ

```



Figure 2-10

In the above example, a list of students applying for admission to a particular graduate class is being prepared according to their grade point averages (GPA's). All the students with GPA greater than or equal to 3.6 are considered bright (denoted by B). Those with GPA less than 3.6, but greater than or equal to 3.0 are considered acceptable (denoted by A). Others are unacceptable (U).

EXAMPLES OF THE SHOW COMMAND

Display of the ASCII collating sequence

```
>DATA IS ASCII, SEQUENCE IS ASCII
>SHOW SEQUENCE
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del
```

Figure 2-11

This command displays the collating sequence determined by the first 128 characters of the ASCII code. The sequence is displayed on the line printer if the OFFLINE parameter is also used.

Display of the EBCDIC collating sequence

```
>DATA A SEQ EBCDIC
>SH S
nul soh stx etx ht del vt ff cr so si dle dc1 dc2 dc3 bs
can em fs gs rs us lf etb esc enq ack bel syn eot dc4 nak
sub sp [ . < ( + ! & ] $ * ) ; ^ -
/ @ , % - > ? ` : # ' = " a b
c d e f g h i j k l m n o p q r
~ s t u v w x y z { A B C D E F
G H I } J K L M N O P Q R \ S T
U V W X Y Z 0 1 2 3 4 5 6 7 8 9
```

Figure 2-12

The EBCDIC collating sequence is displayed if the SHOW command is preceded by the DATA command with the EBCDIC parameter.

The recurring display of the collating sequence

```

>DATA A SEQ A
>SH S
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del

>A MERGE "A-C" WITH "D-L"
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A D B E C F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del

>A "A" = "B"
nul soh stx etx eot enq ack bel bs ht lf vt ff cr so si
dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
sp ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A= D B E C F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ del

>SH NOSEQUENCE
>A MERGE "a-c" WITH "A-C"

```

Figure 2-13

Once specified in the SHOW command, the collating sequence is displayed after each subsequent ALTSEQ command until you specify the NOSEQUENCE parameter.

Sending the collating sequence to a disc file

```
:FILE DISPLOUT=DSPL, NEW; TEMP; DEV=DISC; REC=-80,,F, ASCII
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 12:46 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>DATA A SEQ A
>A "DFT" WITH "ZS"
>SH S, OFFLINE
>EXIT
```

END OF PROGRAM

```
:EDITOR
```

```
HP32201A.7.04 EDIT/3000 SUN, SEP 23, 1979, 12:46 PM
(C) HEWLETT-PACKARD CO. 1978
```

```
/T DSPL, UNN; L ALL, UNN
```

nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	lf	vt	ff	cr	so	si
dle	dc1	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
sp=	T	!	"	#	\$	%	&	'	()	*	+	,	-	.
/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>
?	@	A	B	C	E	G	H	I	J	K	L	M	N	O	P
Q	R	S=	F	U	V	W	X	Y	D=	Z	[\]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	del

Figure 2-14

You can store the contents of the collating sequence in a disc file by using a file equation for the formal designator DISPLOUT with the DEV=DISC parameter and appending DEV=DISC to the file equation.

Displaying the translation table in three forms

```
:FILE DISPLOUT=DSTL, NEW; TEMP; REC=-90,,F, ASCII; DEV=DISC
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 1:05 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>DATA A SEQ A
>A "ZSD" = "L"
>SH T, 0
>EX
```

END OF PROGRAM

:EDITOR

```
HP32201A.7.04 EDIT/3000 SUN, SEP 23, 1979, 1:06 PM
(C) HEWLETT-PACKARD CO. 1978
```

/T DSTL, UNN; L ALL, UNN

TABLE OF ORDINAL VALUE ASSIGNED TO EACH CHARACTER.

!	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
0 !	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
1 !	10 !	11 !	12 !	13 !	14 !	15 !	16 !	17 !	18 !	19 !
2 !	20 !	21 !	22 !	23 !	24 !	25 !	26 !	27 !	28 !	29 !
3 !	30 !	31 !	!sp=32	!!= 33	!#= 34	!\$= 35	!%= 36	!&= 37	!'= 38	!`= 39
4 !	!(= 40	!)= 41	!*= 42	!+= 43	!,= 44	!--= 45	!.= 46	!/= 47	!0= 48	!1= 49
5 !	!2= 50	!3= 51	!4= 52	!5= 53	!6= 54	!7= 55	!8= 56	!9= 57	!:= 58	!:= 59
6 !	!<= 60	!>= 61	!>= 62	!?= 63	!@= 64	!A= 65	!B= 66	!C= 67	!D= 68	!E= 69
7 !	!F= 70	!G= 71	!H= 72	!I= 73	!J= 74	!K= 75	!L= 76	!M= 77	!N= 78	!O= 79
8 !	!P= 80	!Q= 81	!R= 82	!S= 83	!T= 84	!U= 85	!V= 86	!W= 87	!X= 88	!Y= 89
9 !	!Z= 90	![= 91	!\= 92	!] = 93	!^= 94	!_ = 95	!\` = 96	!a= 97	!b= 98	!c= 99
10 !	!d=100	!e=101	!f=102	!g=103	!h=104	!i=105	!j=106	!k=107	!l=108	!m=109
11 !	!n=110	!o=111	!p=112	!q=113	!r=114	!s=115	!t=116	!u=117	!v=118	!w=119
12 !	!x=120	!y=121	!z=122	!{=123	! =124	!}=125	!~=126	!=127	! =128	! =129
13 !	130 !	131 !	132 !	133 !	134 !	135 !	136 !	137 !	138 !	139 !
14 !	140 !	141 !	142 !	143 !	144 !	145 !	146 !	147 !	148 !	149 !
15 !	150 !	151 !	152 !	153 !	154 !	155 !	156 !	157 !	158 !	159 !
16 !	160 !	161 !	162 !	163 !	164 !	165 !	166 !	167 !	168 !	169 !
17 !	170 !	171 !	172 !	173 !	174 !	175 !	176 !	177 !	178 !	179 !
18 !	180 !	181 !	182 !	183 !	184 !	185 !	186 !	187 !	188 !	189 !
19 !	190 !	191 !	192 !	193 !	194 !	195 !	196 !	197 !	198 !	199 !
20 !	200 !	201 !	202 !	203 !	204 !	205 !	206 !	207 !	208 !	209 !
21 !	210 !	211 !	212 !	213 !	214 !	215 !	216 !	217 !	218 !	219 !
22 !	220 !	221 !	222 !	223 !	224 !	225 !	226 !	227 !	228 !	229 !
23 !	230 !	231 !	232 !	233 !	234 !	235 !	236 !	237 !	238 !	239 !
24 !	240 !	241 !	242 !	243 !	244 !	245 !	246 !	247 !	248 !	249 !
25 !	250 !	251 !	252 !	253 !	254 !	255 !				

WHEN PASSED TO SORTINIT, THE TABLE ABOVE IS PRECEDED BY TWO BYTES.
THESE FIRST TWO BYTES CONTAIN A FLAG BYTE OF %000 AND A LENGTH BYTE OF %377
RESPECTIVELY

CONTENTS OF THE ALTSEQ ARRAY FOR PROGRAMMATIC USE (DECIMAL BYTE REPRESENTATION):

0,255,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 32, 69,
70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 32, 84, 85, 86, 87, 88, 89,
76, 91, 92, 93, 94, 95, 96, 97, 98, 99,
100,101,102,103,104,105,106,107,108,109,
110,111,112,113,114,115,116,117,118,119,
120,121,122,123,124,125,126,127,128,129,
130,131,132,133,134,135,136,137,138,139,
140,141,142,143,144,145,146,147,148,149,
150,151,152,153,154,155,156,157,158,159,
160,161,162,163,164,165,166,167,168,169,
170,171,172,173,174,175,176,177,178,179,
180,181,182,183,184,185,186,187,188,189,
190,191,192,193,194,195,196,197,198,199,
200,201,202,203,204,205,206,207,208,209,
210,211,212,213,214,215,216,217,218,219,
220,221,222,223,224,225,226,227,228,229,
230,231,232,233,234,235,236,237,238,239,
240,241,242,243,244,245,246,247,248,249,
250,251,252,253,254,255

CONTENTS OF THE ALTSEQ ARRAY FOR PROGRAMMATIC USE (OCTAL WORD REPRESENTATION):
 X000377,
 X000001,X001003,X002005,X003007,X004011,X005013,X006015,X007017,
 X010021,X011023,X012025,X013027,X014031,X015033,X016035,X017037,
 X020041,X021043,X022045,X023047,X024051,X025053,X026055,X027057,
 X030061,X031063,X032065,X033067,X034071,X035073,X036075,X037077,
 X040101,X041103,X020105,X043107,X044111,X045113,X046115,X047117,
 X050121,X051040,X052125,X053127,X054131,X046133,X056135,X057137,
 X060141,X061143,X062145,X063147,X064151,X065153,X066155,X067157,
 X070161,X071163,X072165,X073167,X074171,X075173,X076175,X077177,
 X100201,X101203,X102205,X103207,X104211,X105213,X106215,X107217,
 X110221,X111223,X112225,X113227,X114231,X115233,X116235,X117237,
 X120241,X121243,X122245,X123247,X124251,X125253,X126255,X127257,
 X130261,X131263,X132265,X133267,X134271,X135273,X136275,X137277,
 X140301,X141303,X142305,X143307,X144311,X145313,X146315,X147317,
 X150321,X151323,X152325,X153327,X154331,X155333,X156335,X157337,
 X160341,X161343,X162345,X163347,X164351,X165353,X166355,X167357,
 X170361,X171363,X172365,X173367,X174371,X175373,X176375,X177377,

Figure 2-15

You should use the OFFLINE parameter of the SHOW TABLE command to send the contents of the table to the disc, tape, or line printer. In this case, the table is printed in three forms. The first form is identical to the table displayed on the terminal. The second form contains the same decimal representation of each character as in the first form; without headings and labels and preceded by two characters. The first character specifies the type of input data and collating sequence (ASCII or EBCDIC). The second character gives the total number of characters in the collating sequence minus one. The third form contains the same information as the second form, except it contains the word representation (in octal).

The NOSEQUENCE parameter

```

>A MERGE "E-T" WITH "e-t"
>SH S
nul soh stx etx eot enq ack bel  bs  ht  lf  vt  ff  cr  so  si
dle dc1 dc2 dc3 dc4 nak syn etb can  em sub esc  fs  gs  rs  us
sp  !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
@  A  B  C  D  E  e  F  f  G  g  H  h  I  i  J
j  K  k  L  l  M  m  N  n  O  o  P  p  Q  q  R
r  S  s  T  t  U  V  W  X  Y  Z  [  \  ]  ^  _
`  a  b  c  d  u  v  w  x  y  z  {  |  }  ~ del

>SH NOSEQUENCE
>A "C" = "S"
>SH S
nul soh stx etx eot enq ack bel  bs  ht  lf  vt  ff  cr  so  si
dle dc1 dc2 dc3 dc4 nak syn etb can  em sub esc  fs  gs  rs  us
sp  !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
@  A  B  D  E  e  F  f  G  g  H  h  I  i  J  j
K  k  L= C  l  M  m  N  n  O  o  P  p  Q  q  R
r  S  s  T  t  U  V  W  X  Y  Z  [  \  ]  ^  _
`  a  b  c  d  u  v  w  x  y  z  {  |  }  ~ del

```

Figure 2-16

The NOSEQUENCE parameter suppresses the display of the collating sequence. However, you can again get the display by specifying the SEQUENCE parameter.

The NOTABLE parameter

```

>DATA A SEQ A
>SH NOT
>A "S" = "D"
>SH T

```

TABLE OF ORDINAL VALUE ASSIGNED TO EACH CHARACTER.

!	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
0 !	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
1 !	10 !	11 !	12 !	13 !	14 !	15 !	16 !	17 !	18 !	19 !
2 !	20 !	21 !	22 !	23 !	24 !	25 !	26 !	27 !	28 !	29 !
3 !	30 !	31 !	!sp=32	!!= 33	!"= 34	!#= 35	!\$= 36	!%= 37	!&= 38	!'= 39
4 !	!(= 40	!)= 41	!*= 42	!+= 43	!,= 44	!--= 45	!.= 46	!/= 47	!0= 48	!1= 49
5 !	!2= 50	!3= 51	!4= 52	!5= 53	!6= 54	!7= 55	!8= 56	!9= 57	!:= 58	!:= 59
6 !	!<= 60	!>= 61	!>= 62	!?= 63	!@= 64	!A= 65	!B= 66	!C= 67	!D= 68	!E= 69
7 !	!F= 70	!G= 71	!H= 72	!I= 73	!J= 74	!K= 75	!L= 76	!M= 77	!N= 78	!O= 79
8 !	!P= 80	!Q= 81	!R= 82	!S= 83	!T= 84	!U= 85	!V= 86	!W= 87	!X= 88	!Y= 89
9 !	!Z= 90	![= 91	!\= 92	!] = 93	!^= 94	!_ = 95	!\` = 96	!a= 97	!b= 98	!c= 99
10 !	!d=100	!e=101	!f=102	!g=103	!h=104	!i=105	!j=106	!k=107	!l=108	!m=109
11 !	!n=110	!o=111	!p=112	!q=113	!r=114	!s=115	!t=116	!u=117	!v=118	!w=119
12 !	!x=120	!y=121	!z=122	!{=123	! =124	!}=125	!~=126	!=127	! 128	! 129
13 !	130 !	131 !	132 !	133 !	134 !	135 !	136 !	137 !	138 !	139 !
14 !	140 !	141 !	142 !	143 !	144 !	145 !	146 !	147 !	148 !	149 !
15 !	150 !	151 !	152 !	153 !	154 !	155 !	156 !	157 !	158 !	159 !
16 !	160 !	161 !	162 !	163 !	164 !	165 !	166 !	167 !	168 !	169 !
17 !	170 !	171 !	172 !	173 !	174 !	175 !	176 !	177 !	178 !	179 !
18 !	180 !	181 !	182 !	183 !	184 !	185 !	186 !	187 !	188 !	189 !
19 !	190 !	191 !	192 !	193 !	194 !	195 !	196 !	197 !	198 !	199 !
20 !	200 !	201 !	202 !	203 !	204 !	205 !	206 !	207 !	208 !	209 !
21 !	210 !	211 !	212 !	213 !	214 !	215 !	216 !	217 !	218 !	219 !
22 !	220 !	221 !	222 !	223 !	224 !	225 !	226 !	227 !	228 !	229 !
23 !	230 !	231 !	232 !	233 !	234 !	235 !	236 !	237 !	238 !	239 !
24 !	240 !	241 !	242 !	243 !	244 !	245 !	246 !	247 !	248 !	249 !
25 !	250 !	251 !	252 !	253 !	254 !	255 !				

WHEN PASSED TO SORTINIT, THE TABLE ABOVE IS PRECEDED BY TWO BYTES.
 THESE FIRST TWO BYTES CONTAIN A FLAG BYTE OF %000 AND A LENGTH BYTE OF %377
 RESPECTIVELY.

Figure 2-17

The NOTABLE parameter suppresses the display of the translation table until you give the >SHOW TABLE command.

EXAMPLES OF THE SORT OPERATION

SORT with the terminal as the output file

```
>INPUT R
>OUTPUT *
>KEY 1, 15
>END
Djilas,           Milovan      sociologist   born 1911
Hammarskjold,    Dag          pacifist      born 1905
K'ung,           Ch'iu       preacher      born 551 B.C.
Khan,            Jenghiz     emperor       born 1167 (?)
Lautreamont,     Comte de    novelist      born 1846
Nijinsky,        Vaslav      dancer        born 1890
Noether,         Emmy        mathematician born 1882
Ortega y Gasset, Jose          philosopher   born 1883
Pirandello,      Luigi       playwright    born 1867
Sen,             Mrinal      movie director born 1923
```

Figure 2-18

In this case, R is the input file and terminal is the output file.

Using the terminal as the input and output file

```
>INPUT *
>OUTPUT *
>KEY 1,2
>END
?GLOBE
?APE
?BANANA
?1234
?2345
?3456
?deaf
?CAPITAL
?:EOD
1234
2345
3456
APE
BANANA
CAPITAL
GLOBE
deaf
```

Note that :EOD terminates the input records.

Figure 2-19

SORT with file equations

```
:FILE INPUT=MAIL1
:FILE OUTPUT=TEST
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 1:50 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>KEY 11, 9
>KEY 1, 10
>E
```

STATISTICS

NUMBER OF RECORDS -	13
NUMBER OF INTERMEDIATE PASSES -	0
SPACE AVAILABLE (IN WORDS) -	11,087
NUMBER OF COMPARES -	52
NUMBER OF SCRATCHFILE IO'S -	8
CPU TIME (MINUTES) -	.00
ELAPSED TIME (MINUTES) -	.04

Figure 2-20

This is an alternative procedure to specify MAIL1 as the input file and TEST as the output file. The FILE commands are used before SORT-MERGE/3000 is accessed. Only the subsystem commands, KEY and END, need be specified in this case.

SORT with cards as the input file

```
:FILE IN; DEV=CARD
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 THU, JAN 24, 1980, 1:16 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>INPUT IN
>OUTPUT MAIL1
>KEY 11, 9
>KEY 1, 10
>E
```

STATISTICS

NUMBER OF RECORDS -	25
RECORD SIZE (IN BYTES) -	80
NUMBER OF INTERMEDIATE PASSES -	0
SPACE AVAILABLE (IN WORDS) -	13,346
NUMBER OF COMPARES -	125
NUMBER OF SCRATCHFILE IO'S -	18
CPU TIME (MINUTES) -	.01
ELAPSED TIME (MINUTES) -	.18

Figure 2-21

The input file, IN, is read from a card reader, sorted, and stored as a disc file.

SORT with multiple input files

```
>INPUT (A, R)
>OUTPUT WORLD
>KEY 31, 15
>E
```

STATISTICS

```
NUMBER OF RECORDS =                20
NUMBER OF INTERMEDIATE PASSES =      0
SPACE AVAILABLE (IN WORDS) =        11,089
NUMBER OF COMPARES =                 95
NUMBER OF SCRATCHFILE IO'S =         14
CPU TIME (MINUTES) =                 .01
ELAPSED TIME (MINUTES) =             .01
RECORD SIZE (IN BYTES) =             72
SCRATCH FILE SIZE (# SECTORS) =      85
```

END OF PROGRAM

:EDITOR

HP32201A.7.04 EDIT/3000 SUN, SEP 23, 1979, 2:01 PM

(C) HEWLETT-PACKARD CO. 1978

/T WORLD, UNN;L ALL, UNN

Clift,	Montgomery	actor	born 1920
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894
Nijinsky,	Vaslav	dancer	born 1890
Khan,	Jenghiz	emperor	born 1167 (?)
Rothstein,	Arnold	gangster	born 1882
Chavez,	Cesar	labor leader	born 1927
Noether,	Emmy	mathematician	born 1882
Sen,	Mrinal	movie director	born 1923
Lautreamont,	Comte de	novelist	born 1846
Hammarskjold,	Dag	pacifist	born 1905
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Crane,	Hart	poet	born 1899
Truman,	Harry	politician	born 1884
K'ung,	Ch'iu	preacher	born 551 B.C.
Joplin,	Janis	singer	born 1943
Djilas,	Milovan	sociologist	born 1911
Chamberlain,	Wilt	sportsman	born 1936
Horse,	Crazy	warrior	born 1848

Figure 2-22

Two files, A and R, are sorted and merged in the same SORT operation. WORLD is the output file.

SORT with only the key fields as the output files

```
>INPUT R
>OUTPUT REST, KEY
>KEY 1, 15
>E
```

STATISTICS

NUMBER OF RECORDS =	10
NUMBER OF INTERMEDIATE PASSES =	0
SPACE AVAILABLE (IN WORDS) =	11,054
NUMBER OF COMPARES =	34
NUMBER OF SCRATCHFILE IO'S =	2
CPU TIME (MINUTES) =	.00
ELAPSED TIME (MINUTES) =	.00
RECORD SIZE (IN BYTES) =	72
SCRATCH FILE SIZE (# SECTORS) =	80

END OF PROGRAM

:EDITOR

HP32201A.7.04 EDIT/3000 SUN, SEP 23, 1979, 2:06 PM

(C) HEWLETT-PACKARD CO. 1978

/T REST, UNN;L ALL, UNN

Djilas,
Hammaraskjold,
K'ung,
Khan,
Lautreamont,
Nijinsky,
Noether,
Ortega y Gasset
Pirandello,
Sen,

Figure 2-23

The OUTPUT command contains the KEY parameter so the output file consists of only the key fields.

SORT with logical record numbers and key fields as the output file

```
>INPUT A
>OUTPUT AMERICAN, NUM, KEY
>KEY 1, 15
>END
```

STATISTICS

NUMBER OF RECORDS =	10
NUMBER OF INTERMEDIATE PASSES =	0
SPACE AVAILABLE (IN WORDS) =	11,054
NUMBER OF COMPARES =	31
NUMBER OF SCRATCHFILE IO'S =	2
CPU TIME (MINUTES) =	.00
ELAPSED TIME (MINUTES) =	.00
RECORD SIZE (IN BYTES) =	72
SCRATCH FILE SIZE (# SECTORS) =	80

```
END OF PROGRAM
:RUN FCOPY.PUB.SYS
```

HP32212A.3.08 FILE COPIER (C) HEWLETT-PACKARD CO. 1978

```
>FROM=AMERICAN; TO; CHAR; OCTAL
```

AMERICAN RECORD 0 (%0, #0)

```
00000: 000000 000004 041550 060555 061145 071154 060551 067054 ....Chamberlain,
00010: 020040 020000
```

AMERICAN RECORD 1 (%1, #1)

```
00000: 000000 000010 041550 060566 062572 026040 020040 020040 ....Chavez,
00010: 020040 020000
```

AMERICAN RECORD 2 (%2, #2)

```
00000: 000000 000002 041554 064546 072054 020040 020040 020040 ....Clift,
00010: 020040 020000
```



AMERICAN RECORD 3 (%3, #3)

00000: 000000 000011 041562 060556 062454 020040 020040 020040Crane,
00010: 020040 020000

AMERICAN RECORD 4 (%4, #4)

00000: 000000 000005 044157 071163 062454 020040 020040 020040Horse,
00010: 020040 020000

AMERICAN RECORD 5 (%5, #5)

00000: 000000 000006 045157 070154 064556 026040 020040 020040Joplin,
00010: 020040 020000

AMERICAN RECORD 6 (%6, #6)

00000: 000000 000001 051157 072150 071564 062551 067054 020040Rothstein,
00010: 020040 020000

AMERICAN RECORD 7 (%7, #7)

00000: 000000 000003 052162 072555 060556 026040 020040 020040Truman,
00010: 020040 020000

AMERICAN RECORD 8 (%10, #8)

00000: 000000 000007 053141 067144 062562 061151 066164 026040Vanderbilt,
00010: 020040 020000

AMERICAN RECORD 9 (%11, #9)

00000: 000000 000000 053551 062556 062562 026040 020040 020040Wiener,
00010: 020040 020000

EOF FOUND IN FROMFILE AFTER RECORD 9

10 RECORDS PROCESSED *** 0 ERRORS

Figure 2-24

Note both NUM and KEY are specified in the OUTPUT command.

SORT with a native language collating sequence

```
>INPUT RDM8566D
>OUTPUT RDM8566S
>KEY 1,25,CHARACTER
>LANGUAGE IS GERMAN
>END
```

```
                STATISTICS
NUMBER OF RECORDS =           50,000
NUMBER OF INTERMEDIATE PASSES =           1
SPACE AVAILABLE (IN WORDS) =       13,797
NUMBER OF COMPARES =       810,905
NUMBER OF SCRATCHFILE IO'S =       25,778
CPU TIME (MINUTES) =           1.97
ELAPSED TIME (MINUTES) =           3.26
RECORD SIZE (IN BYTES) =           25
SCRATCH FILE SIZE (# SECTORS) =       7,140
```

```
>INPUT FRENIN
>OUTPUT FRENOUT
>KEY 1,20, BYTE
>KEY 21,25, CHARACTER
>LANGUAGE IS FRENCH
>END
```

```
                STATISTICS
NUMBER OF RECORDS =           1,023
NUMBER OF INTERMEDIATE PASSES =           0
SPACE AVAILABLE (IN WORDS) =       13,436
NUMBER OF COMPARES =       10,828
NUMBER OF SCRATCHFILE IO'S =           694
CPU TIME (MINUTES) =           .04
ELAPSED TIME (MINUTES) =           .35
RECORD SIZE (IN BYTES) =           80
SCRATCH FILE SIZE (# SECTORS) =       412
```

Figure 2-24a

Note only keys of type CHARACTER are sorted according to the native language collating sequence.

EXAMPLES OF THE MERGE OPERATION

MERGE in interactive mode

```
:RUN MERGE.PUB.SYS

HP32214C.02.02 MERGE/3000 SUN, SEP 21, 1980, 2:29 PM
(C) HEWLETT-PACKARD CO. 1980

>INPUT AMERICAN,REST
>OUTPUT WORLD
>KEY 31, 14
>END
PURGE OLD OUTPUT FILE WORLD.PUB.ACCT ? YES

                STATISTICS

NUMBER OF INPUT FILES =                2
NUMBER OF RECORDS =                    20
SPACE AVAILABLE (IN WORDS) =            11,164
NUMBER OF COMPARES =                    18
CPU TIME (MINUTES) =                    .00
ELAPSED TIME (MINUTES) =                .01
```

Figure 2-25

AMERICAN and REST are the two sorted files and they are merged into the file, WORLD.

MERGE in batch mode

```
:JOB MGR.ACCT
JOB NUMBER = #J10
SUN, SEP 23, 1979, 2:39 PM
HP3000 / MPE III B.00.02

:RUN MERGE.PUB.SYS

HP32214C.02.02 MERGE/3000 SUN, SEP 21, 1980, 2:39 PM
(C) HEWLETT-PACKARD CO. 1980

INPUT MAIL1, MAIL2
OUTPUT AMERICAN
KEY 11, 9; 1, 10
E
OUTPUT FILE CLOSED WITH FILENAME OUTPUTO

                STATISTICS

NUMBER OF INPUT FILES =                2
NUMBER OF RECORDS =                    25
SPACE AVAILABLE (IN WORDS) =            11,161
NUMBER OF COMPARES =                    21
CPU TIME (MINUTES) =                    .00
ELAPSED TIME (MINUTES) =                .01
```


:RUN MERGE.PUB.SYS
PROGRAM TERMINATED IN AN ERROR STATE. (CIERR 976)

```
:EDITOR
END OF PROGRAM
HP32201A.7.04 EDIT/3000  SUN, SEP 23, 1979,  2:44 PM
(C) HEWLETT-PACKARD CO. 1978
T OUTPUT0, UNN; L ALL, UNN
/T OUTPUT0, UNN; L ALL, UNN
PLAINS ANTELOPE 201 OPENSACE AVE BIGCOUNTRY WY 49301 369-732-4821
LOIS ANYONE 6190 COURT ST METROPOLIS NY 20115 619-732-4997
KING ARTHUR 329 EXCALIBUR ST CAMELOT CA 61322 812-200-0100
ALI BABA 40 THIEVES WAY SESAME CO 69142 NONE
BLACK BEAR 47 ALLOVER DR ANYWHERE US 00111 NONE
JOHN BIGTOWN 965 APPIAN WAY METROPOLIS NY 20013 619-407-2314
KNEE BUCKLER 974 FISTICUFF DR PUGILIST ND 04321 976-299-2990
SWASH BUCKLER 497 PLAYACTING CT MOVIE TOWN CA 61497 NONE
ANIMAL CRACKERS 1000 ANYWHERE PL ALLOVER US 00001 001-100-1000
MULE DEER 963 FOREST PL NICECOUNTRY CA 97643 493-900-9000
WHITETAIL DEER 34 WOODSY PL BACKCOUNTRY ME 01341 619-433-4333
JAMES DOE 4193 ANY ST ANYTOWN MD 00133 237-408-7100
JANE DOE 3959 TREEWOOD LN BIGTOWN MA 21843 714-399-4563
PRAIRE DOG 493 ROLLINGHILLS DR OPENSACE ND 24321 992-419-4192
JOHN DOUGHE 239 MAIN ST HOMETOWN MA 26999 714-411-1123
MALLARD DUCK 79 MARSH PL PUDDLE DUCK CA 97432 492-492-4922
JENNA GRANDTR 493 TWENTIETH ST PROGRESSIVE CA 61335 799-191-9191
KARISSA GRANDTR 7917 BROADMOOR WAY BIGTOWN MA 21799 713-244-3717
SNOWSHOE HARE 742 FRIGID WAY COLDSPOT MN 37434 732-732-7320
MOUNTAIN LION 796 KING DR THICKET NM 37643 712-712-7122
SPACE MANN 9999 GALAXY WAY UNIVERSE CA 61239 231-999-9999
SWAMP RABBIT 4444 DAMPPLACE RD BAYOU LA 79999 NONE
NASTY RATTLER 243 DANGER AVE DESERTVILLE CA 87654 828-432-4321
BIGHORN SHEEP 999 MOUNTAIN DR HIGHPLACE CO 34567 776-409-9040
GREY SQUIRREL 432 PLEASANT DR FALLCOLORS MA 14321 619-619-6199
:EOJ
```

Figure 2-26

Note SORT-MERGE/3000 creates the file, OUTPUT0, as the file, AMERICAN, already exists.

MERGE with a native language collating sequence

```
:RUN MERGE.PUB.SYS
```

```
HP32214C.04.00 MERGE/3000 WED, NOV 23, 1983, 2:28 PM (C) HEWLETT-PACKARD CO. 1983
```

```
>INPUT FILE1, FILE2  
>OUTPUT DUTCHSRT  
>KEY 1,25  
>LANGUAGE IS DUTCH  
>END
```

STATISTICS

NUMBER OF INPUT FILES =	2
NUMBER OF RECORDS =	1,030
SPACE AVAILABLE (IN WORDS) =	26,767
NUMBER OF COMPARES =	7
CPU TIME (MINUTES) =	.03
ELAPSED TIME (MINUTES) =	.27
RECORD SIZE (IN BYTES) =	80

Figure 2-27

CALLING SORT FROM A FORTRAN/3000 PROGRAM

SECTION

III

This and the next three sections introduce you to the programmatic use of SORT-MERGE/3000. You can sort one or more files from a FORTRAN/3000 program by using intrinsic calls. These intrinsics (SPL/3000 procedures) are part of SORT-MERGE/3000 and are called by using the SYSTEM INTRINSIC declarations in your program. The various parameters of these intrinsics are used by SORT-MERGE/3000 to perform specific operations.

The SORT program intrinsics

The following is a list of the SORT program intrinsics which reside in the SORTLIB segment of the system segmented library (SL.PUB.SYS):

INTRINSIC	DESCRIPTION
SORTINIT	Initiates the SORT operation.
SORTINPUT	Passes the input records, one at a time, to the SORT program only if the <i>inputfiles</i> parameter is not specified in SORTINIT.
SORTOUTPUT	Signals the beginning of SORT and receives each output record from SORT into an array specified by the <i>record</i> parameter. SORTOUTPUT signals the end of the input process if SORTINPUT is also called. SORTOUTPUT is used only if the <i>ouputfiles</i> parameter of SORTINIT is not specified.
SORTEND	Closes the scratch file and restores the data stack to its original state. It signals the beginning of SORT if SORTOUTPUT is not called.
SORTSTAT	Prints the SORT statistics on \$STDLIST.
SORTTITLE	Prints the version number and title of the SORTLIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.
SORTERRORMESS	Called to retrieve and print a message if a fatal error occurs during SORT. SORTERRORMESS is called from a user supplied error procedure (the <i>errorproc</i> parameter of SORTINIT).

The call to SORTINIT starts the SORT operation. You should follow it by calling SORTINPUT if the *inputfiles* parameter of SORTINIT is not specified. After this, call SORTOUTPUT if the *outputfiles* parameter of SORTINIT is not specified. Then call SORTEND to terminate SORT. SORTINIT and SORTEND must be called from the same procedure. If you want the display of the SORT statistics, call SORTSTAT. Additionally, call SORTERRORMESS from the user supplied procedure, *errorproc*, if you want a display of the message when an error occurs. Note SORTINIT and SORTEND are always required. But the calls to SORTINPUT, SORTOUTPUT, SORTSTAT, and SORTERRORMESS are optional. However, their order is important whenever they are called. Optional intrinsic SORTTITLE is an exception in that it can be called from the program at any stage after the declaration of the system intrinsics. The following flowchart describes the SORT operation when SORTINPUT, SORTOUTPUT, and SORTSTAT are used:

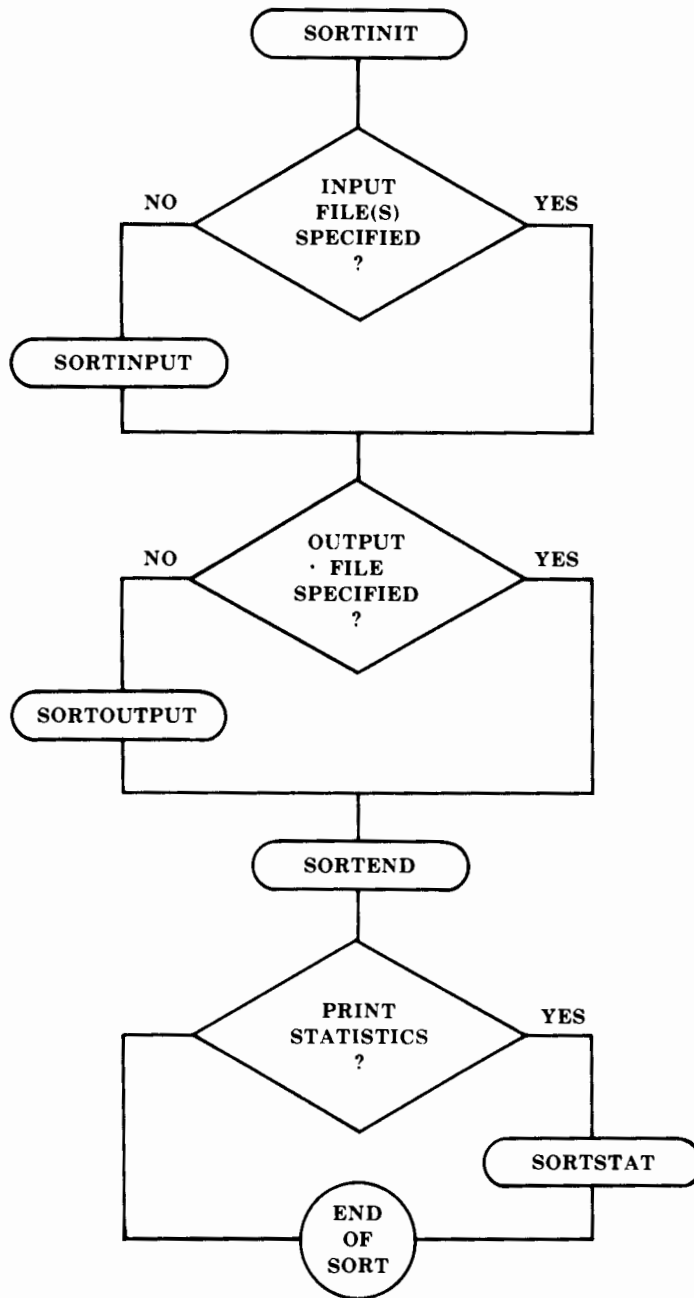


Figure 3-1. Flowchart of SORT Intrinsic

Preparation and Execution of the SORT Programs

The amount of available memory can affect both the time required to perform the SORT operation and the amount of secondary storage needed by a temporary file. SORT programs should normally be prepared with the maximum available segsize, which is specified by the MAXDATA= segsize parameter of the :PREP or :RUN commands.

If a smaller segsize must be used, the following allowances should be made to provide enough space for sorting. The segsize should be approximately 12000 words greater than the space required to run your program without calls to the SORT intrinsics. If the files to be sorted are opened NOBUF, the segsize should be increased by an additional amount equal to your blocksize (in words). When sorting files that have been opened multirecord, the maximum possible segsize should be used.

If the error message INSUFFICIENT STACK SPACE is displayed, increase the MAXDATA parameter. If the message TOO MANY FILES OPEN (FSERR 71) appears, it means MPE has no room for its tables in the user data segment. Use the NOCB parameter of the :RUN command during the execution of the program in this case.

NOTE: The SORTINITIALF intrinsic is included in this manual for the maintenance of existing FORTRAN/3000 programs.

SORTINIT

Initiates the SORT operation.

SYNTAX

```
CALL SORTINIT ( IA inputfiles, IA outputfiles, IV outputoption, IV reclen,  
DV numrecs, IV numkeys, IA keys, IA altseq, LP keycompare,  
P errorproc, IA statistics, L failure,  
I errorparm, I spaceallocation, IA charseq, O-V parm1)
```

PARAMETERS

- inputfiles* An integer array containing the MPE/3000 file identification numbers (fnum's) of the files to be sorted. The array must be terminated with a word of zero to indicate the end of the list. If the files are opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking. \$Null is not a valid input file.
- outputfiles* An integer array containing the file identification of the output file. The second word must contain a zero to indicate the end of the list. If the file is opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking.
- outputoption* An integer which determines the format of the output records. There are four possibilities:
- 0—Output record is the same as input record (default value)
 - 1—Output record is a double integer (4 characters) whose value is the logical (relative) record number of the record.
 - 2—Output record contains only the key fields, concatenated together with the major keys on the left followed by the remaining keys (Fig. 3-2).
 - 3—Output record is the logical record number followed by the key fields.
- reclen* An integer which denotes the maximum length of a record in characters. If it is not specified, the record length is taken from the first file specified in the *inputfiles* array. In this case, you must specify the *inputfiles* parameter.

numrecs A double integer which is the upper bound to the number of records to be sorted. If this is not specified (or if all the input files are not on the disc), the value of 10,000 (double integer) is used. Otherwise, the parameter value is derived from the file label (the end of file number of the input files).

numkeys and *keys* *numkeys* is an integer and *keys* is an integer array. They specify the way the records are sorted. If either is specified, the other must also be specified and the *keycompare* parameter must not be specified. *numkeys* is the number of keys used in the comparison of records and must be either equal to or greater than one. For each key being specified, *keys* contains three words:

First word gives the position of the first character of the key within the input record. (The first character of the record is considered position 1.) Second word gives the total number of characters in the key. Third word (bits 0 through 7) gives the ordering sequence of the records; 0 for ascending, 1 for descending. bits 8 through 15 of the third word indicate the type of data according to the following convention:

0=logical or byte (same as the *type*, BYTE, in interactive mode)

1=two's complement (including integer and double integer)

2=floating point (including real and long)

3=packed decimal

5=packed decimal with even number of digits

4=Display-Trailing-Sign (see the KEY command in Section II)

6=Display-Leading-Sign

7=Display-Leading-Sign-Separate

8=Display-Trailing-Sign-Separate

9=Character (Collating sequence of *charseq* [see below] is used)

NOTE: The integrity of the integer array *keys* must be maintained throughout the sorting operation. Do not change the array until after a call has been made to the intrinsic SORTEND.

altseq

An integer array defining an alternate collating sequence. The first character (bits 0-7) of the array is defined according to the following table:

Sequence	ASCII	EBCDIC	ALTSEQ
Data			
ASCII	255	2	0
EBCDIC	1	255	undefined

Table 3-1. The First Character of the *altseq* Array

The second character (bits 8-15) specifies one less than the total number of characters in the collating sequence (in this case, 255 or %377). These two characters are followed by the actual collating sequence responsible for the particular SORT operation. See Figs. 3-2 through 3-5 for details.

keycompare

A user-supplied logical function subprogram that allows you access to your records when they are compared (Fig. 3-6). It must be specified in your call to SORTINIT if you do not specify *numkeys* and *keys*. This subprogram should include a statement of the following form:

LOGICAL FUNCTION *keycompare* (*rec1*, *len1*, *rec2*, *len2*)

rec1 and *rec2* are pointers to the two records and *len1* and *len2* are the lengths of the records in characters. The subprogram returns a true value if *rec1* precedes *rec2*, and a false value otherwise. A true value is also returned in the case of ties, to ensure the records with equal keys retain their original order.

errorproc

A user-supplied subroutine subprogram called whenever a fatal error occurs during a SORT operation. It is used along with the SORTERRORMESS intrinsic and should include a statement of the following form:

SUBROUTINE *errorproc*(*errorcode*)

errorcode is an integer which is the SORT program error number. It is passed to *errorproc* when an error occurs. See Figs. 3-8 and 3-9. If *errorproc* or *errorparm* are not specified, a default procedure is used which displays the error message corresponding to the particular *errorcode*. For a list of these error messages, see Appendix A.

statistics

An integer array which, if specified, is filled with the following data (Fig. 3-9):

First and second words=
number of records sorted (double integer)

Third word=
number of intermediate passes

Fourth word=
space available for sorting

Fifth and sixth words=
number of comparisons (double integer)

Seventh and eighth words=
number of scratch file inputs/outputs (double integer)

Ninth and tenth words=
CPU time used (in milliseconds, double integer)

Eleventh and twelfth words=
elapsed time (in milliseconds, double integer)

failure A logical variable, which if specified, is set to -1 (true) if a fatal error occurs, and 0 (false) otherwise. It is set after each call to SORTINPUT and SORTOUTPUT; in addition, the condition code is set (Fig. 3-14).

Error conditions:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

errorparm An integer variable which, if specified, is set to the SORTLIB error number if an error occurs. The SORTERRORMESS intrinsic can be used to obtain the error message text. If the errorparm is supplied, the errorproc procedure is ignored and no error messages are displayed. For a list of error messages see Appendix A.

spaceallocation An integer variable which, if specified, is used to determine stack allocation. A positive spaceallocation specifies the number of words that may be used for sorting and buffering. A negative value specifies the number of words that should be left for the user after determining the amount available. Zero will cause a default value of 2500 words left for the user to be used.

NOTE: Internal SORT routines, which may call procedures such as FOPEN (typically requiring about 850 stack words), could cause a system error. The user is warned to be careful when overriding this default value.

charseq 2 word integer array. Set the first word to 1. Set the second word to the language id of the native language whose collating sequence is to be used to sort keys of type 9 (character).

parm1 unused

SORTINPUT

Passes the input records, one at a time, to the SORT program, only if the *inputfiles* parameter is not specified in SORTINIT.

SYNTAX

```
CALL SORTINPUT ( LA record, IV length )
```

PARAMETERS

<i>record</i>	A logical array containing a data record.
<i>length</i>	An integer denoting the number of characters in the record. It should be long enough to contain all the keys specified, but not longer than the record size (<i>reclen</i>).

ERROR CONDITIONS:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

This intrinsic follows SORTINIT and precedes SORTOUTPUT and SORTEND (see Fig. 3-1).

SORTOUTPUT

Signals the beginning of SORT and receives each output record from SORT into an array specified by the *record* parameter. SORTOUTPUT signals the end of the input process if SORTINPUT is also called. SORTOUTPUT is used only if the *outputfiles* parameter of SORTINIT is not specified.

SYNTAX

LA *I*

CALL SORTOUTPUT (*record*, *length*)

PARAMETERS

<i>record</i>	A logical array receiving the next output record in the format specified by <i>outputoption</i> .
<i>length</i>	An integer passed by reference denoting the number of characters returned in the record. When no more records remain, <i>length</i> is set to -1.

ERROR CONDITIONS:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

Note if SORTINPUT is also called, SORTOUTPUT is called only after SORTINPUT has passed all the records. SORTOUTPUT always precedes SORTEND (Fig. 3-11).

SORTEND

Closes the scratch file and restores the data stack to its original state. It signals the beginning of SORT if SORTOUTPUT is not called.

SYNTAX

CALL SORTEND

ERROR CONDITIONS:

CCE=

no error occurred during SORT (*failure set to false*)

CCL=

an error occurred during SORT (*failure set to true*)

This intrinsic is required if SORTINIT is called. It can be called either after all the calls to the output file are completed by SORTINIT, or after all the calls to SORTOUTPUT are completed.

■ **Note:** SORTEND must be called by the same subroutine that called SORTINIT.

SORTSTAT

Prints the SORT statistics on \$STDLIST.

SYNTAX

```
CALL SORTSTAT ( IA
                 statistics )
```

statistics is an integer array. SORTSTAT is called after SORTEND (Fig. 3-12).

SORTTITLE

Prints the version number and title of the SORTLIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.

SYNTAX

CALL SORTTITLE

It can be called from the program at any stage after the declaration of the system intrinsics (Fig. 3-13).

SORTERRORMESS

Called to retrieve and print a message if a fatal error occurs during SORT. SORTERRORMESS is called from a user supplied procedure (the *errorproc* parameter of SORTINIT).

SYNTAX

```
CALL SORTERRORMESS ( IV      BA      I  
                    (errorcode, message, length)
```

PARAMETERS

<i>errorcode</i>	An integer (the SORT program error number) passed to <i>errorproc</i> when an error occurs.
<i>message</i>	A character array into which the text of the message is placed. The <i>message</i> parameter must be at least 72 characters long.
<i>length</i>	An integer denoting the length of the message in characters.

SORTERRORMESS works in conjunction with the *errorproc* parameter of SORTINIT (Fig. 3-8).

SORTINITIALF

Initiates the SORT operation (to be used only for existing FORTRAN/3000 programs).

SYNTAX

```
CALL SORTINITIALF ( IV inputfile, IV outputfile, IV outputoption, IV reclen,  
                   DV numrecs, IV numkeys, IA keys, P errorproc, LP keycompare,  
                   LA statistics, L failure ) O-V
```

PARAMETERS

- inputfile* MPE/3000 file number of the file to be sorted. Input records are read directly from the file by the SORT program, and no calls are made to SORTINPUT. If *inputfile* is not specified, the records are passed via SORTINPUT which must be called.
- outputfile* MPE/3000 file number of the file to which sorted records are sent. If specified, no calls to SORTOUTPUT may be made. Otherwise, the sorted records are sent through the SORTOUTPUT intrinsic which must be called.

Unlike SORTINIT, where the *inputfiles* and *outputfiles* parameters are arrays, the parameters, *inputfile* and *outputfile*, are integers; each of them representing only a single file. SORTINITIALF intrinsic does not have the capability of defining an alternate collating sequence. Also, the positions of the *errorproc* and *keycompare* parameters are interchanged. The remaining parameters follow the same rules as in SORTINIT (Fig. 3-14).

EXAMPLES

Calling the SORTINIT intrinsic when both *inputfiles* and *outputfiles* are specified

```
$CONTROL USLINIT, FILE=33,FILE=34,FILE=35
PROGRAM F1
CHARACTER *72 BUF
INTEGER KEYS(6),FNUM,INFILE(3),OUTFILE(2)
SYSTEM INTRINSIC SORTINIT, SORTEND

C
C SORT THE FILES, A (FTN33) AND R (FTN34), INTO A FILE, WORLD (FTN35).
C SORT ON LAST NAMES WITHIN OCCUPATIONS.
C ESTABLISH THE KEYS. MAJOR AT 31 (OCCUPATION) FOR 17 BYTES AND
C THE OTHER KEY AT 1 (LAST NAME) FOR 15 BYTES.
C
KEYS(1)=31
KEYS(2)=17
KEYS(3)[0:8]=1
KEYS(3)[8:8]=0
KEYS(4)=1
KEYS(5)=17
KEYS(6)[0:8]=1
KEYS(6)[8:8]=0

C
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
INFILE(1)= FNUM(33)
INFILE(2)=FNUM(34)
INFILE(3)=0
OUTFILE(1)=FNUM(35)
OUTFILE(2)=0

C
C INITIALIZE SORT - OUTPUTOPTION=2
C
CALL SORTINIT(INFILE,OUTFILE,2,,2,KEYS)
CALL SORTEND

C
C READ AND DISPLAY THE KEY FIELDS.
C
REWIND 35
10 READ(35,END=100)BUF
DISPLAY BUF
GO TO 10
100 STOP
END
```



```
:FILE FTN33=A, OLD
:FILE FTN34=R, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS;MAXDATA=4000
```

END OF PREPARE

warrior	Horse,
sportsman	Chamberlain,
sociologist	Djilas,
singer	Joplin,
preacher	K'ung,
politician	Truman,
poet	Crane,
playright	Pirandello,
philosopher	Ortega y Gasset,
pacifist	Hammarskjold,
novelist	Lautreamont,
movie director	Sen,
mathematician	Noether,
labor leader	Chavez,
gangster	Rothstein,
emperor	Khan,
dancer	Nijinsky,
cybernetician	Wiener,
capitalist	Vanderbilt,
actor	Clift,

Figure 3-2

The files, A and R, are sorted and merged into the file, WORLD. INFILE and OUTFILE are the *inputfiles* and *outputfiles* parameters. The default value is used for the *reclen* parameter. The *outputoption* is two which sends only the key fields to the output file. The third word of the array KEYS has one in the first eight bits, which accounts for a descending sequence; and zero in the last eight bits, which specifies the BYTE data type.

Construction of the translation table

```
:FILE DISPLOUT=DSFL, NEW; TEMP; REC=-110,,F, ASCII; DEV=DISC
:RUN SORT.PUB.SYS
```

```
HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 3:10 PM
(C) HEWLETT-PACKARD CO. 1980
```

```
>DATA A SEQ A
>A "BA" = "AB"
>SH T, 0
>EX
```

Figure 3-3

Note the ALTSEQ command allocates a higher value to B than A in the altered collating sequence. The >SHOW TABLE, OFFLINE command stores the translation table on a disc as the file equation is appended by DEV=DISC. The contents of the table are copied into the file, DSFL.

Editing the contents of the translation table

```
/T DSFL, UNN; L ALL, UNN
```

TABLE OF ORDINAL VALUE ASSIGNED TO EACH CHARACTER.

!	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
0 !	0 !	1 !	2 !	3 !	4 !	5 !	6 !	7 !	8 !	9 !
1 !	10 !	11 !	12 !	13 !	14 !	15 !	16 !	17 !	18 !	19 !
2 !	20 !	21 !	22 !	23 !	24 !	25 !	26 !	27 !	28 !	29 !
3 !	30 !	31 !	!sp=32	!!= 33	!="= 34	!#= 35	!\$= 36	!%= 37	!&= 38	!'= 39
4 !	(= 40	!)= 41	!*= 42	!+= 43	!,= 44	!-= 45	!.= 46	!/= 47	!0= 48	!1= 49
5 !	!2= 50	!3= 51	!4= 52	!5= 53	!6= 54	!7= 55	!8= 56	!9= 57	!:= 58	!:= 59
6 !	!<= 60	!>= 61	!>= 62	!?= 63	!@= 64	!A= 66	!B= 65	!C= 67	!D= 68	!E= 69
7 !	!F= 70	!G= 71	!H= 72	!I= 73	!J= 74	!K= 75	!L= 76	!M= 77	!N= 78	!O= 79
8 !	!P= 80	!Q= 81	!R= 82	!S= 83	!T= 84	!U= 85	!V= 86	!W= 87	!X= 88	!Y= 89
9 !	!Z= 90	![= 91	!\= 92	!] = 93	!^= 94	!_ = 95	!\` = 96	!a= 97	!b= 98	!c= 99
10 !	!d=100	!e=101	!f=102	!g=103	!h=104	!i=105	!j=106	!k=107	!l=108	!m=109
11 !	!n=110	!o=111	!p=112	!q=113	!r=114	!s=115	!t=116	!u=117	!v=118	!w=119
12 !	!x=120	!y=121	!z=122	!{=123	! =124	!}=125	!~=126	! =127	! 128	! 129
13 !	130 !	131 !	132 !	133 !	134 !	135 !	136 !	137 !	138 !	139 !
14 !	140 !	141 !	142 !	143 !	144 !	145 !	146 !	147 !	148 !	149 !
15 !	150 !	151 !	152 !	153 !	154 !	155 !	156 !	157 !	158 !	159 !
16 !	160 !	161 !	162 !	163 !	164 !	165 !	166 !	167 !	168 !	169 !
17 !	170 !	171 !	172 !	173 !	174 !	175 !	176 !	177 !	178 !	179 !
18 !	180 !	181 !	182 !	183 !	184 !	185 !	186 !	187 !	188 !	189 !
19 !	190 !	191 !	192 !	193 !	194 !	195 !	196 !	197 !	198 !	199 !
20 !	200 !	201 !	202 !	203 !	204 !	205 !	206 !	207 !	208 !	209 !
21 !	210 !	211 !	212 !	213 !	214 !	215 !	216 !	217 !	218 !	219 !
22 !	220 !	221 !	222 !	223 !	224 !	225 !	226 !	227 !	228 !	229 !
23 !	230 !	231 !	232 !	233 !	234 !	235 !	236 !	237 !	238 !	239 !
24 !	240 !	241 !	242 !	243 !	244 !	245 !	246 !	247 !	248 !	249 !
25 !	250 !	251 !	252 !	253 !	254 !	255 !				

WHEN PASSED TO SORTINIT, THE TABLE ABOVE IS PRECEDED BY TWO BYTES.

THESE FIRST TWO BYTES CONTAIN A FLAG BYTE OF %000 AND A LENGTH BYTE OF %377 RESPECTIVELY.

32	
33	CONTENTS OF THE ALTSEQ ARRAY FOR PROGRAMMATIC USE (DECIMAL BYTE REPRESENTATION):
34	0,255,
35	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
36	10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
37	20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
38	30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
39	40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
40	50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
41	60, 61, 62, 63, 64, 66, 65, 67, 68, 69,
42	70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
43	80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
44	90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
45	100,101,102,103,104,105,106,107,108,109,
46	110,111,112,113,114,115,116,117,118,119,
47	120,121,122,123,124,125,126,127,128,129,
48	130,131,132,133,134,135,136,137,138,139,
49	140,141,142,143,144,145,146,147,148,149,
50	150,151,152,153,154,155,156,157,158,159,
51	160,161,162,163,164,165,166,167,168,169,
52	170,171,172,173,174,175,176,177,178,179,
53	180,181,182,183,184,185,186,187,188,189,
54	190,191,192,193,194,195,196,197,198,199,
55	200,201,202,203,204,205,206,207,208,209,
56	210,211,212,213,214,215,216,217,218,219,
57	220,221,222,223,224,225,226,227,228,229,
58	230,231,232,233,234,235,236,237,238,239,
59	240,241,242,243,244,245,246,247,248,249,
60	250,251,252,253,254,255

```

61
62     CONTENTS OF THE ALTSEQ ARRAY FOR PROGRAMMATIC USE (OCTAL WORD
        REPRESENTATION):
63     %000377,
64     %000001,%001003,%002005,%003007,%004011,%005013,%006015,%007017,
65     %010021,%011023,%012025,%013027,%014031,%015033,%016035,%017037,
66     %020041,%021043,%022045,%023047,%024051,%025053,%026055,%027057,
67     %030061,%031063,%032065,%033067,%034071,%035073,%036075,%037077,
68     %040102,%040503,%042105,%043107,%044111,%045113,%046115,%047117,
69     %050121,%051123,%052125,%053127,%054131,%055133,%056135,%057137,
70     %060141,%061143,%062145,%063147,%064151,%065153,%066155,%067157,
71     %070161,%071163,%072165,%073167,%074171,%075173,%076175,%077177,
72     %100201,%101203,%102205,%103207,%104211,%105213,%106215,%107217,
73     %110221,%111223,%112225,%113227,%114231,%115233,%116235,%117237,
74     %120241,%121243,%122245,%123247,%124251,%125253,%126255,%127257,
75     %130261,%131263,%132265,%133267,%134271,%135273,%136275,%137277,
76     %140301,%141303,%142305,%143307,%144311,%145313,%146315,%147317,
77     %150321,%151323,%152325,%153327,%154331,%155333,%156335,%157337,
78     %160341,%161343,%162345,%163347,%164351,%165353,%166355,%167357,
79     %170361,%171363,%172365,%173367,%174371,%175373,%176375,%177377,
80
/DQ 1/62, 80
NUMBER OF LINES DELETED = 63
/CQ 1 TO "      " IN ALL
/CQ 1/6 TO "    *** IN ALL
K DSFL, UNN
DSFL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD? YES
/E

```

Figure 3-4

The file DSFL is edited with the help of EDIT/3000. Note the lines 1 through 62 and 80 are deleted. Lines 63 through 79 are moved to the seventh column positions before inserting DSFL into the array, ALT, in the FORTRAN program.

Calling SORTINIT with the *altseq* parameter

```
$CONTROL USLINIT,FILE=31,FILE=32
PROGRAM F2
INTEGER KEYS(6),FNUM,IFILE(2),OFILE(2),ALT(129)
SYSTEM INTRINSIC SORTINIT,SORTEND
CHARACTER *72 SHOW
DATA ALT/
*%000377,
*%000001,%001003,%002005,%003007,%004011,%005013,%006015,%007017
*%010021,%011023,%012025,%013027,%014031,%015033,%016035,%017037
*%020041,%021043,%022045,%023047,%024051,%025053,%026055,%027057
*%030061,%031063,%032065,%033067,%034071,%035073,%036075,%037077
*%040102,%040503,%042105,%043107,%044111,%045113,%046115,%047117
*%050121,%051123,%052125,%053127,%054131,%055133,%056135,%057137
*%060141,%061143,%062145,%063147,%064151,%065153,%066155,%067157
*%070161,%071163,%072165,%073167,%074171,%075173,%076175,%077177
*%100201,%101203,%102205,%103207,%104211,%105213,%106215,%107217
*%110221,%111223,%112225,%113227,%114231,%115233,%116235,%117237
*%120241,%121243,%122245,%123247,%124251,%125253,%126255,%127257
*%130261,%131263,%132265,%133267,%134271,%135273,%136275,%137277
*%140301,%141303,%142305,%143307,%144311,%145313,%146315,%147317
*%150321,%151323,%152325,%153327,%154331,%155333,%156335,%157337
*%160341,%161343,%162345,%167747,%164351,%165353,%166355,%167357
*%170361,%171363,%172365,%173367,%174371,%175373,%152325,%177377
```

```
C
C SORT THE FILE, UNDRGRAD (FTN31), INTO A FILE, VICTORS(FTN32).
C SORT ON LAST NAMES WITHIN GRADES.
C ESTABLISH THE KEYS. MAJOR AT 38 FOR 1 BYTE AND
C MINOR AT 1 FOR 3 BYTES.
```

```
C
C KEYS(1)=38
C KEYS(2)=1
C KEYS(3)=0
C KEYS(4)=1
C KEYS(5)=3
C KEYS(6)=0
```

```
C
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
```

```
C
C IFILE(1)=FNUM(31)
C IFILE(2)=0
C OFILE(1)=FNUM(32)
C OFILE(2)=0
```

```
C
```

```

C   INITIALIZE SORT USING THE altseq PARAMETER, ALT.
C
      CALL SORTINIT(IFILE,OFILE,,,,2,KEYS,ALT)
      CALL SORTEND
C
C   READ AND DISPLAY THE OUTPUT FILE.
C
      REWIND 32
11  READ(32,END=100)SHOW
      DISPLAY SHOW
      GO TO 11
100 STOP
      END

```

```

:FILE FTN31=UNDRGRAD, OLD
:FILE FTN32=VICTORS, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

Nicolas Bourbaki	4.0	B
Milind Ranade	3.9	B
Sensible Kommunist	3.6	B
Uncle Samuelson	3.7	B
Vegetarian Dracula	3.8	B
Boris Frankenstein	3.1	A
Hit Woman	3.1	A
Homo Genius	3.4	A
Lacy Lowercase	3.4	A
Red Butler	3.1	A
Sorting Jack	3.3	A
Tech Nitpicker	3.2	A
Virgin Cat	3.1	A
Harry Krishna	2.9	U
Thomas Collins	2.1	U

Figure 3-5

Calling SORTINIT with EBCDIC as the collating sequence

```

$CONTROL USLINIT,FILE=29,FILE=30
PROGRAM F3
INTEGER KEYS(6),FNUM,AL(129),IFILE(2),OFILE(2)
SYSTEM INTRINSIC SORTINIT,SORTEND
CHARACTER *72 BUF

C
C SORT THE FILE, NAMES (FTN29), INTO A FILE, ARRANGED (FTN30).
C SORT ON LAST NAMES
C ESTABLISH THE KEY AT 1 FOR 3 BYTES
C
KEYS(1)=1
KEYS(2)=3
KEYS(3)=0

C
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
IFILE(1)=FNUM(29)
IFILE(2)=0
OFILE(1)=FNUM(30)
OFILE(2)=0

C
C DESCRIBES THE AL ARRAY.
C
DATA AL/
*%001377,
*%000001,%001003,%033455,%027057,%013005,%022413,%006015,%007017
*%010021,%011023,%036075,%031046,%014031,%037447,%021035,%032437
*%040132,%077573,%055554,%050175,%046535,%056116,%065540,%045541
*%170361,%171363,%172365,%173367,%174371,%075136,%046176,%067157
*%076301,%141303,%142305,%143307,%144311,%150722,%151724,%152726
*%153730,%154742,%161744,%162746,%163750,%164655,%160275,%057555
*%074601,%101203,%102205,%103207,%104211,%110622,%111624,%112626
*%113630,%114642,%121644,%122646,%123650,%124700,%065320,%045007
*%100201,%101203,%102205,%103207,%104211,%105213,%106215,%107217
*%110221,%111223,%112225,%113227,%114231,%115233,%116235,%117237
*%120241,%121243,%122245,%123247,%124251,%125253,%126255,%127257
*%130261,%131263,%132265,%133267,%134271,%135273,%136275,%137277
*%140301,%141303,%142305,%143307,%144311,%145313,%146315,%147317
*%150321,%151323,%152325,%153327,%154331,%155333,%156335,%157337
*%160341,%161343,%162345,%167747,%164351,%165353,%166355,%167357
*%170361,%171363,%172365,%173367,%174371,%175373,%152325,%177377

C
```

```

C   INITIALIZE SORT USING THE altseq PARAMETER, AL.
C
      CALL SORTINIT(IFILE,OFILE,,,,1,KEYS,AL)
      CALL SORTEND
C
C   READ AND DISPLAY THE OUTPUT FILE.
C
      REWIND 30
10  READ(30,END=100 )BUF
      DISPLAY BUF
      GO TO 10
100 STOP
      END

:FILE FTN29=NAMES, OLD
:FILE FTN30=ARRANGED, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

END OF PREPARE

villon
winthrop
wylie
yamakoshi
ziegfeld
zydner
BRADLEY
COMTE
CONNINGHAM
CONNOR
CORDAN
DELIUS

END OF PROGRAM

```

Figure 3-6

The >SHOW TABLE, OFFLINE command preceded by the appropriate file equation and the DATA command with the EBCDIC sequence parameter, copies the translation table to the disc file. The file is edited and the *altseq* array is constructed in the same manner as in the previous example. The file, NAMES, is sorted into the file, ARRANGED. Note the lowercase alphabetic letters precede the uppercase letters in ARRANGED.

Using the *keycompare* parameter

```
$CONTROL USLINIT,FILE=31,FILE=32
PROGRAM F4
INTEGER FNUM,IN(2),OU(2)
CHARACTER BUF*72
INTEGER L1,L2
EXTERNAL KEYCOM
LOGICAL FUNCTION KEYCOM
SYSTEM INTRINSIC SORTINIT,SORTEND

C
C SORT THE FILE UNGRAD(FTN31) INTO A FILE VICTORS(FTN32).
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
      IN(1)=FNUM(31)
      IN(2)=0
      OU(1)=FNUM(32)
      OU(2)=0

C
C INITIALIZE SORT WITH THE KEYCOMPARE PARAMETER, KEY.
C
      CALL SORTINIT(IN,OU,,,,,,,,KEYCOM)
      CALL SORTEND

C
C READ AND DISPLAY THE OUTPUT FILE.
C
      REWIND 32
500 READ(32,END=100)BUF
      DISPLAY BUF
      GO TO 500
100 STOP
      END
```

PROGRAM UNIT F4 COMPILED

```
LOGICAL FUNCTION KEYCOM(A,B,C,D)
CHARACTER A*(B)
INTEGER B,D
CHARACTER C*(D)
KEYCOM=.FALSE.
IF (A.LE.C)KEYCOM=.TRUE.
RETURN
END
```

PROGRAM UNIT KEYCOM COMPILED

```
:FILE FTN31=UNDRGRAD, OLD
:FILE FTN32=VICTORS, NEW
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

Boris Frankenstein	3.1	A
Harry Krishna	2.9	U
Hit Woman	3.1	A
Homo Genius	3.4	A
Lacy Lowercase	3.4	A
Milind Ranade	3.9	B
Nicolas Bourbaki	4.0	B
Red Butler	3.1	A
Sensible Kommunist	3.6	B
Sorting Jack	3.3	A
Tech Nitpicker	3.2	A
Thomas Collins	2.1	U
Uncle Sammuelson	3.7	B
Vegetarian Dracula	3.8	B
Virgin Cat	3.1	A

Figure 3-7

The *keycompare* parameter, KEYCOM, is specified and the file, UNDRGRAD, is sorted into the file, VICTORS. The major key is established at column one.

Using the *errorproc* parameter without the occurrence of an error

```

$CONTROL USLINIT,FILE=33,FILE=27
  PROGRAM F5
  INTEGER KEYS(6),FNUM,IFILE(2),OFILE(2)
  CHARACTER *72 BUF
  EXTERNAL ERROR
  SYSTEM INTRINSIC SORTINIT, SORTEND
C
C SORT THE FILE, A (FTN33), INTO THE FILE, AMERICAN
C (FTN27). SORT ON LAST NAMES WITHIN OCCUPATIONS.
C ESTABLISH THE KEYS. MAJOR AT 31 (OCCUPATION) FOR 17 BYTES
C AND MINOR AT 1 (LAST NAME) FOR 15 BYTES.
C
  KEYS(1)=31
  KEYS(2)=17
  KEYS(3)=0
  KEYS(4)=1
  KEYS(5)=15
  KEYS(6)=0
C

```



```

C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
  IFILE(1)=FNUM(33)
  IFILE(2)=0
  OFILE(1)=FNUM(27)
  OFILE(2)=0
C
C INITIALIZE SORT WITH ERRORPROC PARAMETER, ERROR.
C
  CALL SORTINIT(IFILE,OFILE,,,,2,KEYS,,,ERROR)
  CALL SORTEND
C
C READ AND DISPLAY THE OUTPUT FILE.
C
  REWIND 27
10 READ(27,END=100)BUF
  DISPLAY BUF
  GO TO 10
100 STOP
  END

```

PROGRAM UNIT F5 COMPILED

```

  SUBROUTINE ERROR(ERRORCODE)
  INTEGER ERRORCODE
  CHARACTER *72 MESSAGE
  SYSTEM INTRINSIC SORTERRORMESS
  MESSAGE=""
C
C CALL THE SYSTEM INTRINSIC, SORTERRORMESS.
C
  CALL SORTERRORMESS(ERRORCODE,MESSAGE;L)
C
C DISPLAY ERROR MESSAGE AND NUMBER IF THERE IS ANY
C FATAL ERROR.
C
  DISPLAY MESSAGE
  RETURN
  END

```

PROGRAM UNIT ERROR COMPILED

```

:FILE FTN33=A, OLD
:FILE FTN27=AMERICAN, NEW
:PREPRUN $OLDPASS; MAXDATA=15000
END OF PREPARE

```

Clift,	Montgomery	actor	born 1920
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894
Rothstein,	Arnold	gangster	born 1882
Chavez,	Cesar	labor leader	born 1927
Crane,	Hart	poet	born 1899
Truman,	Harry	politician	born 1884
Joplin,	Janis	singer	born 1943
Chamberlain,	Wilt	sportsman	born 1936
Horse,	Crazy	warrior	born 1848

Figure 3-8

Using the *errorproc* parameter during the occurrence of an error

```

$CONTROL USLINIT,FILE=33,FILE=27
PROGRAM F6
INTEGER KEYS(6),FNUM,IFILE(2),OFILE(2)
CHARACTER *72 BUF
EXTERNAL ERROR
SYSTEM INTRINSIC SORTINIT,SORTEND
C
C SORT THE FILE, A (FTN33), INTO THE FILE, AMERICAN
C (FTN27). SORT ON LAST NAMES WITHIN OCCUPATIONS.
C ESTABLISH THE KEYS. MAJOR AT 0 (OCCUPATION) FOR 17 BYTES
C AND MINOR AT 0 (LAST NAME) FOR 15 BYTES.
C
      KEYS(1)=0
      KEYS(2)=17
      KEYS(3)=0
      KEYS(4)=0
      KEYS(5)=15
      KEYS(6)=0
C
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
      IFILE(1)=FNUM(33)
      IFILE(2)=0
      OFILE(1)=FNUM(27)
      OFILE(2)=0
C

```

```

C   INITIALIZE SORT WITH THE ERRORPROC PARAMETER, ERROR.
C
      CALL SORTINIT(IFILE,OFILE,,,,2,KEYS,,,ERROR)
      CALL SORTEND
C
C   READ AND DISPLAY OUTPUT FILE.
C
      REWIND 27
10  READ(27,END=100)BUF
      DISPLAY BUF
      GO TO 10
100 STOP
      END

```

PROGRAM UNIT F6 COMPILED

```

      SUBROUTINE ERROR(ERRORCODE)
      INTEGER ERRORCODE
      CHARACTER *72 MESSAGE
      SYSTEM INTRINSIC SORTERRORMESS
      MESSAGE=""
C
C   CALL SYSTEM INTRINSIC SORTERRORMESS
C
      CALL SORTERRORMESS(ERRORCODE,MESSAGE,L)
C
C   DISPLAY ERROR MESSAGE AND NUMBER IF THERE IS ANY
C   FATAL ERROR.
C
      DISPLAY MESSAGE
      RETURN
      END

```

PROGRAM UNIT ERROR COMPILED

```

:FILE FTN33=A, OLD
:FILE FTN27=AMERICAN, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

END OF PREPARE

```

SORTLIB: KEYFIELD IS NOT WITHIN SPECIFIED RECORD LENGTH

Figure 3-9

Note even though the *errorproc* parameter, **ERROR**, is specified in the first case, the file, **UNDRGRAD**, is sorted into the file, **VICTORS**, as there is no occurrence of an error. The error occurs in the second case when the keys are purposely specified at the column positions zero. This prevents the **SORT** operation from being performed. The **SORTLIB** message, **KEYFIELD IS NOT WITHIN SPECIFIED RECORD LENGTH**, appears on the terminal.

Displaying the SORT statistics with the *statistics* parameter

```

$CONTROL USLINIT,FILE=33,FILE=27
PROGRAM F7
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,IN(2),OUT(2),STAT(12),NOPAS,SPACE
INTEGER *4 NOREC, NOCOMP, NOSCFLE,CPTME,ELTME
SYSTEM INTRINSIC SORTINIT,SORTEND
EQUIVALENCE (NOPAS,STAT(3)),(SPACE,STAT(4))
EQUIVALENCE (NOREC,STAT(1)),(NOCOMP,STAT(5)),(NOSCFLE,STAT(7))
EQUIVALENCE (CPTME,STAT(9)),(ELTME,STAT(11))
C
C SORT THE FILE, A (FTN33), INTO THE FILE, AMERICAN (FTN27).
C ESTABLISH THE KEYS. MAJOR AT 1 FOR 15 BYTES
C
KEYS(1)=1
KEYS(2)=15
KEYS(3)=0
C
C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
IN(1)=FNUM(33)
IN(2)=0
OUT(1)=FNUM(27)
OUT(2)=0
C
C INITIALIZE SORT WITH THE STAT PARAMETER.
C
CALL SORTINIT(IN,OUT,,,,1,KEYS,,,,STAT)
CALL SORTEND
DISPLAY"THE STATISTICS OF THE SORT OPERATION ARE:"
DISPLAY"      "
DISPLAY NOREC,NOPAS,SPACE,NOCOMP,NOSCFLE,CPTME,ELTME
DISPLAY" "
C
C READ AND DISPLAY THE OUTPUT FILE.
C
REWIND 27
10 READ(27,END=100)BUF
DISPLAY BUF
GO TO 10
100 STOP
END
```

```
:FILE FTN33=A, OLD
:FILE FTN34=AMERICAN, NEW
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

THE STATISTICS OF THE SORT OPERATION ARE:

	10	0	11596	31	8	170	365
Chamberlain,	Wilt		sportsman	born	1936		
Chavez,	Cesar		labor leader	born	1927		
Clift,	Montgomery		actor	born	1920		
Crane,	Hart		poet	born	1899		
Horse,	Crazy		warrior	born	1848		
Joplin,	Janis		singer	born	1943		
Rothstein,	Arnold		gangster	born	1882		
Truman,	Harry		politician	born	1884		
Vanderbilt,	Cornelius		capitalist	born	1794		
Wiener,	Norbert		cybernetician	born	1894		

Figure 3-10

The array, STAT, is displayed and is followed by the output file, AMERICAN.

Calling SORTINPUT

```

$CONTROL USLINIT,FILE=33,FILE=34,FILE=35
PROGRAM F8
CHARACTER *72 BUF
INTEGER KEYS(6),FNUM,OUTFILE(2)
SYSTEM INTRINSIC SORTINIT,SORTEND,SORTINPUT
EQUIVALENCE(BUF,LBUF)
LOGICAL LBUF(36)

C
C
C SORT THE FILES, A (FTN33) AND R (FTN34), INTO THE FILE,
C WORLD (FTN35).
C ESTABLISH THE KEYS.
C
KEYS(1)=1
KEYS(2)=15
KEYS(3)=0
KEYS(4)=31
KEYS(5)=17
KEYS(6)=0

C
C ESTABLISH NUMBERS FOR THE OUTPUT FILE.
C
OUTFILE(1)=FNUM(35)
OUTFILE(2)=0

C
C INITIALIZE SORT WITH THE RECLLEN PARAMETER BUT NO
C INPUTFILES PARAMETER.
C
CALL SORTINIT(,OUTFILE,,72,,2,KEYS)
C
```

```

C   CALL THE SORTINPUT INTRINSIC TO READ THE FILE, A.
C
50  READ(33,END=100)BUF
    CALL SORTINPUT(LBUF,72)
    GO TO 50
C
C   CALL THE SORTINPUT INTRINSIC TO READ THE FILE, W.
C
100 READ(34,END=200)BUF
    CALL SORTINPUT(LBUF,72)
    GO TO 100
200 CALL SORTEND
C
C   READ AND DISPLAY THE OUTPUT FILE.
C
    REWIND 35
    10 READ(35,END=300)BUF
        DISPLAY BUF
        GO TO 10
    300 STOP
        END

```



```

:FILE FTN33=A, OLD
:FILE FTN34=R, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

Chamberlain,	Wilt	sportsman	born 1936
Chavez,	Cesar	labor leader	born 1927
Clift,	Montgomery	actor	born 1920
Crane,	Hart	poet	born 1899
Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
Horse,	Crazy	warrior	born 1848
Joplin,	Janis	singer	born 1943
K'ung,	CH'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Rothstein,	Arnold	gangster	born 1882
Sen,	Mrinal	movie director	born 1923
Truman,	Harry	politician	born 1884
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894

Figure 3-11

SORTINPUT is called since the *inputfiles* parameter is not specified in the call to SORTINIT.

Calling SORTINIT with the *charseq* parameter

```

$CONTROL USLINIT,FILE=31,FILE=32
PROGRAM F13
INTEGER KEYS(3),FNUM,IFILE(2),OFILE(2),CSEQ(2),ERR
SYSTEM INTRINSIC SORTINIT,SORTEND
CHARACTER *72 SHOW

C
C   SORT THE FILE, MISCDATA (FTN31) INTO A FILE, SORTDATA (FTN32)
C   SORT ON THE FIRST 24 CHARACTERS USING THE COLLATING SEQUENCE
C   FOR THE NATIVE LANGUAGE, AMERICAN
C
C
C   ESTABLISH THE KEY
C
C   KEYS(1)=1
C   KEYS(2)=24
C   KEYS(3)=9
C
C   ESTABLISH THE CHARSEQ PARAMETER; LANGUAGE=AMERICAN;LANGUAGE ID=
C
C   CSEQ(1)=1
C   CSEQ(2)=1
C
C   ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES
C
C   IFILE(1)=FNUM(31)
C   IFILE(2)=0
C   OFILE(1)=FNUM(32)
C   OFILE(2)=0
C
C   INITIALIZE SORT USING THE charseq PARAMETER
C
C   CALL SORTINIT(IFILE,OFILE,,,1,KEYS,,,,,ERR,,CSEQ)
C   IF (ERR.NE.0) DISPLAY ERR
C   CALL SORTEND
C
C   READ AND DISPLAY THE OUTPUT FILE
C
C   REWIND 32
11  READ(32,END=100)SHOW
    DISPLAY SHOW
    GO TO 11
100 STOP
    END
```

PROGRAM UNIT F13 COMPILED

```
****      GLOBAL STATISTICS      ****
**** NO ERRORS, NO WARNINGS ****
TOTAL COMPILATION TIME  0:00:01
TOTAL ELAPSED TIME     0:00:02
TOTAL SYMBOL TABLE USED  % 07
```

```
END OF COMPILE
:PREP $OLDPASS,PF13;MAXDATA=20000
```

```
END OF PREPARE
:FILE FTN31=MIUSCDATA,old
:FILE FTN32=SORTDATA,new
:RUN PF13
```

```
aardvark
Chicago
Elève
Essen
étude
mañana
München
straße
Troy
Zoological Gardens
¿cuantos?
```

Figure 3-11a

Calling SORTOUTPUT

```
#CONTROL USLINIT,FILE=34,FILE=28
PROGRAM F9
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,INFILES(2)
LOGICAL LBUF(36)
EQUIVALENCE(LBUF,BUF)
SYSTEM INTRINSIC SORTINIT,SORTOUTPUT,SORTEND

C
C SORT THE FILE, R, INTO THE FILE, REST. ESTABLISH THE
C KEYS. ESTABLISH NUMBERS FOR THE INPUT FILE.
C
    INFILES(1)=FNUM(34)
    INFILES(2)=0
    KEYS(1)=1
    KEYS(2)=10
    KEYS(3)=0

C
C INITIALIZE SORT WITHOUT THE OUTPUTFILES PARAMETER.
C
    CALL SORTINIT(INFILES,,,,,1,KEYS)

C
C CALL THE SORTOUTPUT INTRINSIC.
C
50 CALL SORTOUTPUT(LBUF,LEN)
   IF(LEN.LE.-1)GO TO 60
   DISPLAY BUF
   GO TO 50
60 CONTINUE
   CALL SORTEND
   STOP
   END
```

```
:FILE FTN34=R, OLD
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Sen,	Mrinal	movie director	born 1923

Figure 3-12

The *outputfiles* parameter is not specified and the file, A, is sorted into the file, AMERICAN.

Calling SORTSTAT

```
$CONTROL USLIMIT,FILE=34,FILE=28
PROGRAM F10
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,IN(2),OUT(2),STATISTICS(12)
C
C SORT THE FILE, R, INTO THE FILE, REST.
C ESTABLISH THE KEYS AND NUMBERS FOR THE FILES.
C
SYSTEM INTRINSIC SORTINIT,SORTEND,SORTSTAT
KEYS(1)=1
KEYS(2)=10
KEYS(3)=0
IN(1)=FNUM(34)
IN(2)=0
OUT(1)=FNUM(28)
OUT(2)=0
C
C INITIALIZE SORT.
C
CALL SORTINIT(IN,OUT,,,,1,KEYS,,,,STATISTICS)
CALL SORTEND
C
C CALL THE SORTSTAT INTRINSIC.
C
CALL SORTSTAT(STATISTICS)
C
C READ AND DISPLAY THE OUTPUT FILE.
C
REWIND 28
10 READ(28,END=100)BUF
DISPLAY BUF
GO TO 10
100 STOP
END

:FILE FTN34=R, OLD
:FILE FTN28=REST,NEW
:PREPRUN $OLDPASS; MAXDATA=15000

END OF PREPARE
```

STATISTICS

NUMBER OF RECORDS =			10
NUMBER OF INTERMEDIATE PASSES =			0
SPACE AVAILABLE (IN WORDS) =		11,603	
NUMBER OF COMPARES =			34
NUMBER OF SCRATCHFILE IO'S =			8
CPU TIME (MINUTES) =			.00
ELAPSED TIME (MINUTES) =			.01
Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Sen,	Mrinal	movie director	born 1923

Figure 3-13

Notice the SORT statistics are printed in a more useful format compared to the case when the *statistics* parameter is specified in SORTINIT.

Calling SORTTITLE

```

$CONTROL USLINIT,FILE=33,FILE=34,FILE=35
  PROGRAM F11
  CHARACTER *72 BUF
  INTEGER KEYS(6),FNUM,INFILE(3),OUTFILE(2)
  SYSTEM INTRINSIC SORTINIT,SORTEND,SORTTITLE
C
C  SORT THE FILES, A AND R, INTO THE FILE, WORLD.
C  ESTABLISH THE KEYS AND NUMBERS FOR THE FILES.
C
  KEYS(1)=1
  KEYS(2)=15
  KEYS(3)=0
  KEYS(4)=31
  KEYS(5)=17
  KEYS(6)=0
  INFILE(1)= FNUM(33)
  INFILE(2)=FNUM(34)
  INFILE(3)=0
  OUTFILE(1)=FNUM(35)
  OUTFILE(2)=0
C

```

```

C   INITIALIZE SORT.
C
      CALL SORTINIT(INFILE,OUTFILE,,,,2,KEYS)
      CALL SORTEND
C
C   CALL THE SORTTITLE INTRINSIC.
C
      CALL SORTTITLE
C
C   READ AND DISPLAY THE OUTPUT FILE.
C
      REWIND 35
10  READ(35,END=100)BUF
      DISPLAY BUF
      GO TO 10
100 STOP
      END

```

```

:FILE FTN33=A, OLD
:FILE FTN34=R, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

HP32214C.02.02 SORT/3000 SUN, SEP 21, 1980, 6:34 PM
(C) HEWLETT-PACKARD CO. 1980

Chamberlain,	Wilt	sportsman	born 1936
Chavez,	Cesar	labor leader	born 1927
Clift,	Montgomery	actor	born 1920
Crane,	Hart	poet	born 1899
Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
Horse,	Crazy	warrior	born 1848
Joplin,	Janis	singer	born 1943
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Rothstein,	Arnold	gangster	born 1882
Sen,	Mrinal	movie director	born 1923
Truman,	Harry	politician	born 1884
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894

Figure 3-14

Note the display of the version number and title of the SORTLIB segment along with the date and time produced by the DATELINE intrinsic.

Calling SORTINITIALF with the *failure* parameter

```

$CONTROL USLINIT,INIT,FILE=21,FILE=22
PROGRAM F12
CHARACTER*9 BUF,NAME
INTEGER KEYS(3),FNUM
LOGICAL FAILURE,LBUF(5)
EQUIVALENCE (LBUF,BUF)
SYSTEM INTRINSIC SORTINITIALF,SORTOUTPUT,SORTEND
10  FORMAT(//T20," NAME",6X,"NUMBER"/)
20  FORMAT(T20,S,T30,I3)
30  FORMAT(//T25,"TOTAL = ",I3////)
C
C PRINT A SORTED REPORT OF ALL THE LAST NAMES IN MAIL1,
C THE NUMBER OF TIMES A NAME APPEARS,
C AND THE TOTAL NUMBER OF NAMES IN THE FILE.
C
C PRINT HEADING
C
WRITE(6,10)
C
C ESTABLISH KEY
C
KEYS(1)=11
KEYS(2)=9
KEYS(3)=0
C
C INITIALIZE SORT - OUTPUT OPTION = 2
C OUTPUT = KEY FIELD ONLY
C
CALL SORTINITIALF(FNUM(21),,2,,
#1,KEYS,,,FAILURE)
IF(FAILURE)STOP 100
50  CALL SORTOUTPUT(LBUF,LEN)
IF(LEN.EQ.-1)GO TO 500
IF(FAILURE)STOP 200
C
C REPORT GENERATION SECTION
C
ITOTAL=ITOTAL+1
IF(BUF.EQ.NAME)GO TO 60
IF(ICTR.EQ.0)GO TO 70
WRITE(6,20)NAME,ICTR
70  NAME=BUF
ICTR=0
60  ICTR=ICTR+1
GO TO 50
500 WRITE(6,30)ITOTAL
C
C END OF REPORT GENERATION
C
CALL SORTEND
IF(FAILURE)STOP 300
STOP
END
```

```

:FILE FTN21=MAIL2, OLD
:PREPRUN $OLDPASS; MAXDATA=15000

END OF PREPARE

```

NAME	NUMBER
ANYONE	1
ARTHUR	1
BABA	1
BIGTOWN	1
BUCKLER	2
DOE	2
DOUGHE	1
GRANDTR	2

TOTAL = 12

Figure 3-15

The key fields are read from the input file, MAIL2, in the sorted order for printing a report.

Multirecord, NOBUF, and Buffered Files

```

:FORTRAN NOBUF 1

```

```

PAGE 0001 HP32102B.01.02 FORTRAN/3000 (C) HEWLETT-PACKARD CO. 1979 TUE, JAN

```

```

$CONTROL FILE=10-13,USLINIT
PROGRAM PNOBUF 1

```

```

C
C
C
C
C
C
C
C

```

```

This program demonstrates the use of multirecord, NOBUF,
and buffered files with the SORT intrinsic. The failure
parameter is checked and the errorparm is used to obtain
the error message text if needed.

```

```

SYSTEM INTRINSIC SORTINIT, SORTEND, SORTSTAT, SORTERRORMESS
INTEGER INPUT(4), OUTPUT(2), KEYS(3), STATS(12)
INTEGER ERROR, FNUM, MSGLEN
LOGICAL FAILED
CHARACTER*72 MSG

```

```

C
C
C

```

```

Establish the file numbers for the input and output files

```

```

INPUT(1) = FNUM(11)
INPUT(2) = FNUM(12)
INPUT(3) = FNUM(13)
INPUT(4) = 0

```

```

C

```

```

OUTPUT(1) = FNUM(10)
OUTPUT(2) = 0

```

```

C      Keys to sort the sequence numbers in columns 73-80
C
      KEYS(1) = 73
      KEYS(2) = 8
      KEYS(3) = 0
C
      FAILED = .FALSE.
      ERROR = 0
C
C      Sort the files.
C
      CALL SORTINIT(INPUT,OUTPUT,0,80,,1,KEYS,,,,STATS,FAILED,ERROR)
      CALL SORTEND
C
C      Print error message if one occurred, otherwise statistics.
C
      IF (FAILED) GO TO 100
C
      CALL SORTSTAT(STATS)
      GOTO 200
100    CALL SORTERRORMESS(ERROR,MSG,MSGLEN)
      DISPLAY MSG[1:MSGLEN],`` ( '',ERROR,`` )''
200    STOP
      END

```

PROGRAM UNIT PNOBUF1 COMPILED

PAGE 0002 HEWLETT-PACKARD 32102B.01.02 FORTRAN/3000 TUE, JAN 29, 1980, 11

```

****      GLOBAL STATISTICS      ****
****      NO ERRORS, NO WARNINGS  ****
TOTAL COMPILATION TIME  0:00:02
TOTAL ELAPSED TIME      0:00:04

```

END OF COMPILE

:PREP \$OLDPASS,PNOBUF1;MAXDATA=31232

END OF PREPARE

```

:COMMENT                                INPUT FILES
:FILE FTN11=UNSORT01,OLD;MR
:FILE FTN12=UNSORT02,OLD;NOBUF
:FILE FTN13=UNSORT03,OLD
:COMMENT                                OUTPUT FILE
:FILE FTN10=SORTED,NEW;MR;SAVE
:RUN PNOBUF1;LIB=G

```

STATISTICS

```

NUMBER OF RECORDS =                150
NUMBER OF INTERMEDIATE PASSES =      0
SPACE AVAILABLE (IN WORDS) =        27,588
NUMBER OF COMPARES =                1,170
NUMBER OF SCRATCHFILE IO'S =         102
CPU TIME (MINUTES) =                 .02
ELAPSED TIME (MINUTES) =             .40

```

END OF PROGRAM

CALLING MERGE FROM A FORTRAN/3000 PROGRAM

SECTION

IV

You can merge two or more sorted files from a FORTRAN/3000 program by calling the MERGE program intrinsics. These intrinsics (SPL/3000 procedures) are part of SORT-MERGE/3000 and are called by using the SYSTEM INTRINSIC declarations in your program. The various parameters of these intrinsics are used by SORT-MERGE/3000 to perform specific operations.

The MERGE program intrinsics

The following is a list of the MERGE program intrinsics which reside in the MERGELIB segment of the system segmented library:

INTRINSIC	DESCRIPTION
MERGEINIT	Merges two or more sorted files.
MERGEOUTPUT	Requests records from MERGEINIT, one at a time, if the <i>outputfiles</i> parameter is not specified in MERGEINIT.
MERGEEND	Restores the data stack to its original state. MERGEEND must be called only if MERGEINIT is called.
MERGESTAT	Prints the MERGE statistics on \$STDLIST.
MERGETITLE	Prints the version number and title of the MERGELIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.
MERGEERRORMESS	Called to retrieve and print a message if a fatal error occurs during MERGE. Called from a user supplied error procedure (the <i>errorproc</i> parameter of MERGEINIT).



The MERGEINIT intrinsic initiates the MERGE operation. After calling MERGEINIT, you should call MERGEOUTPUT if the *outputfiles* parameters of MERGEINIT is not specified. This is followed by a call to the MERGEEND intrinsic. MERGEINIT and MERGEEND must be called from the same procedure. Call MERGESTAT if you want the display of the MERGE statistics. Additionally, call MERGEERRORMESS from the user supplied procedure, *errorproc*, if you want a display of the message when an error occurs. The calls to the intrinsics, MERGEOUTPUT, MERGESTAT, and MERGEERRORMESS, are optional but the order is important if they are called. The optional intrinsic, MERGETITLE, can be called at any stage. The following flowchart illustrates the MERGE operation when MERGEOUTPUT and MERGESTAT are called:

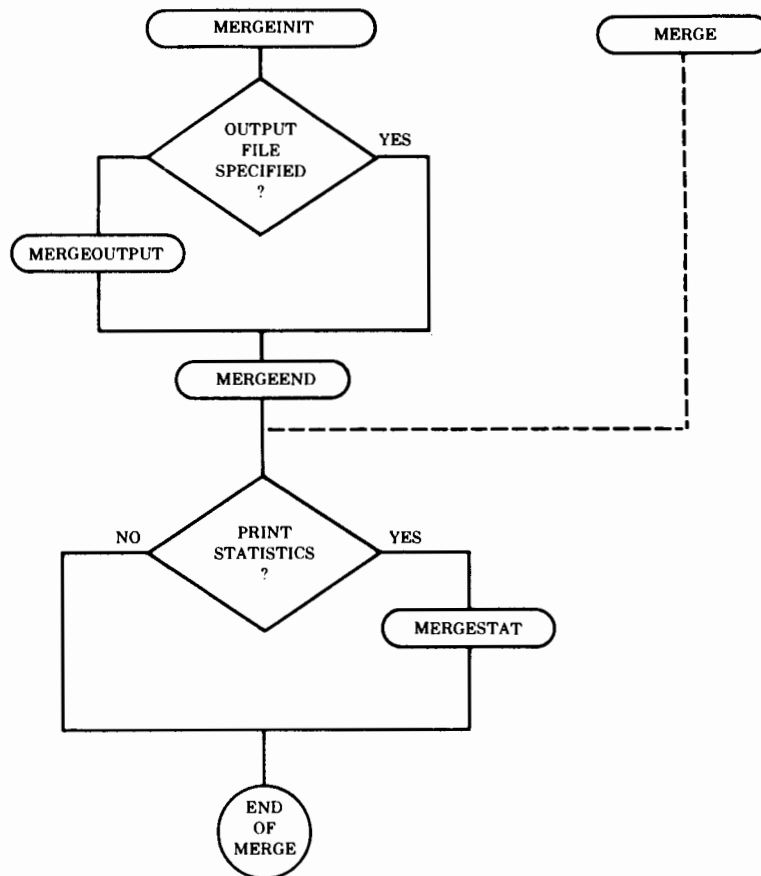


Figure 4-1. Flowchart of Merge Intrinsics

Preparation and Execution of the MERGE programs

The amount of available memory can affect the time required to perform a MERGE. MERGE programs with files opened multirecord should be prepared with the maximum available segsize which is specified by the MAXDATA=segsize parameter of the :PREP or :RUN commands. MERGE programs with files opened NOBUF should increase the segsize, allowing one block per file.

If the error message INSUFFICIENT STACK SPACE is displayed, increase the MAXDATA parameter. If the message TOO MANY FILES OPEN (FSERR 71) appears, it means MPE has no room for its tables in the user data segment. Use the NOCB parameter of the :RUN command during the execution of the program in this case.

NOTE: The MERGE intrinsic is included in this manual for the maintenance of existing FORTRAN/3000 programs.

MERGEINIT

Merges two or more sorted files.

SYNTAX

```
CALL MERGEINIT ( IAinputfiles, Ppreprocessor, IAoutputfiles,
Ppostprocessor, LVkeyonly, IVnumkeys, IAkeys, IAaltseq,
LPkeycompare, Perrorproc, IAstatistics, Lfailure,
Ierrorparm, Ispaceallocation, IAcharseq, O-Vparm1)
```

PARAMETERS

- inputfiles* An integer array containing the file identifications (fnum's) of the input files to be merged. The array is terminated with a word of zero. If the files are opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking. \$Null is not a valid input file.
- preprocessor* A subroutine which is called whenever a record is read from the input files (Fig. 4-1). The call should include a statement of the following form:
- ```
SUBROUTINE preprocessor (file, record, length)
```
- file* is an integer denoting the index to the *inputfiles* array of the file from which the record is read. Its value lies between 0 and the number of input files minus one. *record* is a character array denoting the data record. *length* is an integer denoting the number of characters in the record.
- outputfiles* An integer array containing the file identification of the output file. The second word must contain a zero to indicate the end of the list. If the file is opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking.
- postprocessor* A subroutine which is called before each record is sent to the output file. Either this parameter or *outputfiles* (or both) must be specified in any programmatic MERGE operation. The call should include a statement of the following form:
- ```
SUBROUTINE postprocessor (record, length)
```
- record* is a character array specifying the data record. *length* is an integer denoting the number of characters in the record.
- keyonly* A logical variable, which if true, causes only the key fields; concatenated together with the major key on the left followed by the remaining keys; to be sent as output (Fig. 4-3). The *keycompare* parameter must not be specified in this case. If *keyonly* is false, the entire record is sent as output (Fig. 4-4). The default for *keyonly* is false.

numkeys and *keys*

numkeys is an integer and *keys* is an integer array. They describe the way records are merged. If either is specified, the other must also be specified and *keycompare* must not be specified. *numkeys* is the number of keys used during the comparison of records. It may be either equal to or greater than one. *keys* specifies the way records are compared. For each key being specified, *keys* contains three words:

First word gives the position of the first character of the key within the record. Second word gives the number of characters in the key. Third word (bits 0 through 7) gives the ordering sequence of the records. (0 for ascending, 1 for descending) bits 8 through of the third word 15 indicate the type of data according to the following convention:

0= logical or byte (same as the *type*, BYTE, in interactive mode)

1=two's complement (including integer and double integer)

2=floating point

3=packed decimal

5=packed decimal with even number of digits

4=Display-Trailing-Sign (see the KEY command in Section II)

6=Display-Leading-Sign

7= Display-Leading-Sign-Separate

8= Display-Trailing-Sign-Separate

9= Character (collating sequence of *charseq* [see below] is used)

Note: The integrity of the integer array *keys* must be maintained throughout the merge operation. Do not change it until after a call has been made to the intrinsic MERGEEND.

altseq

An integer array defining an alternate collating sequence. The first character (bits 0-7) of the array is defined according to Table 3-1.

The second character (bits 8-15) specifies one less than the total number of characters in the collating sequence (in this case, 255 or %377). These two characters are followed by the actual collating sequence responsible for the particular MERGE operation.

keycompare

A user-supplied function subprogram which must be specified if you do not specify *numkeys* and *keys*. It is called whenever two records are compared. The call should include a statement of the following form:

LOGICAL FUNCTION *keycompare* (*rec1*, *len1*, *rec2*, *len2*)

rec1 and *rec2* are pointers to the two records and *len1* and *len2* are the lengths of the records in characters. *keycompare* returns a true value if *rec1* precedes *rec2*, and a false value otherwise. It returns a true value even in the case of ties. This ensures that the original sequence is preserved in the case of ties.

<i>errorproc</i>	<p>A user-supplied subroutine, which is used in conjunction with the MERGEERRORMESS intrinsic. It is called as follows whenever a fatal error occurs during the MERGE operation:</p> <p>SUBROUTINE <i>errorproc</i>(<i>errorcode</i>)</p> <p><i>errorcode</i> is an integer which is the MERGE program error number. It is passed to <i>errorproc</i> when an error occurs. If <i>errorproc</i> or <i>errorparm</i> are not specified, a default procedure is used which prints an error message corresponding to the particular <i>errorcode</i>. For a list of these errors, see Appendix A.</p>
<i>statistics</i>	<p>An integer array which, if specified, is filled with the following data (Fig. 4-5):</p> <p>First word= number of input files.</p> <p>Second and third words= number of merged records (double integer)</p> <p>Fourth word= space available for merging.</p> <p>Fifth and sixth words= number of comparisons (double integer).</p> <p>Seventh and eighth words= CPU time (in milliseconds, double integer).</p> <p>Ninth and tenth words= elapsed time (in milliseconds, double integer).</p>
<i>failure</i>	<p>A logical variable which, if specified, is set to -1 (true) if a fatal error occurs, and 0 (false) otherwise (Fig. 4-10).</p> <p>Error conditions:</p> <p>CCE= no error occurred (<i>failure</i> set to false)</p> <p>CCG= an error occurred (<i>failure</i> set to true)</p>
<i>errorparm</i>	<p>An integer variable which, if specified, is set to the MERGELIB error number if an error occurs. The MERGEERRORMESS intrinsic can be used to obtain the error message text. If the <i>errorparm</i> is supplied, the <i>errorproc</i> procedure is ignored and no error messages are displayed. For a list of error messages see Appendix A.</p>

spaceallocation An integer variable which, if specified, is used to determine stack allocation. A positive *spaceallocation* specifies the number of words that may be used for sorting and buffering. A negative values specifies the number of words that should be left for the user after determining the amount available. Zero will cause a default value to be used.

charseq 2 word integer array. Set the first word to 1. Set the second word to *langnum* of the native language whose collating sequence is to be used to sort keys of type 9 (character).

parm1 unused

MERGEOUTPUT

Requests records from MERGEINIT, one at a time, if the *outputfiles* parameter is not specified in MERGEINIT.

SYNTAX

```
CALL MERGEOUTPUT ( LA, I
                   record, length )
```

PARAMETERS

<i>record</i>	A logical array receiving the next output record.
<i>length</i>	An integer denoting the number of characters in the record.

It can be called from the program after MERGEINIT (Fig. 4-6).

MERGEEND

Restores the data stack to its original state.

SYNTAX

CALL MERGEEND

It is called only if MERGEINIT is called, and it must be called from the same subroutine that called MERGEINIT.

MERGESTAT

Prints the MERGE statistics on \$STDLIST.

SYNTAX

```
CALL MERGESTAT (IAstatistics)
```

statistics is an integer array. See Fig. 4-7 for details.

MERGETITLE

Prints the version number and title of the MERGELIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.

SYNTAX

CALL MERGETITLE

MERGETITLE can be called from the program at any stage after the declaration of the system intrinsics (Fig. 4-8).

MERGEERRORMESS

Called to retrieve and print a message if a fatal error occurs during MERGE. MERGEERRORMESS is called from a user supplied error procedure (the *errorproc* parameter of MERGEINIT).

SYNTAX

```
CALL MERGEERRORMESS ( IV errorcode, BA message, I length )
```

PARAMETERS

<i>errorcode</i>	An integer (MERGE program error number) passed to <i>errorproc</i> when an error occurs.
<i>message</i>	A character array into which the text of the message is placed. The <i>message</i> parameter must be at least 72 characters long.
<i>length</i>	An integer denoting the length of the message in characters.

MERGEERRORMESS converts MERGEINIT error code values into ASCII strings. It works in conjunction with the *errorproc* parameter of MERGEINIT (Fig. 4-9).

MERGE

Initiates the MERGE program (to be used only for existing FORTRAN/3000 programs).

SYNTAX

```
CALL MERGE ( IV numinputfiles, IA inputfiles, IV outputfile, IV keyonly,  
IV numkeys, IA keys, P preprocessor, P postprocessor,  
P errorproc, LP keycompare, IA statistics, L failure ) O-V
```

PARAMETERS

<i>numinputfiles</i>	An integer denoting the number of input files to be merged. This parameter is not optional and is either equal to or greater than one.
<i>inputfiles</i>	An integer array containing the MPE/3000 file numbers of the files to be merged. These file numbers appear in the locations <i>inputfiles</i> (0) through <i>inputfiles</i> (<i>numinputfiles</i> -1). <i>inputfiles</i> is not an optional parameter.
<i>outputfile</i>	Unlike MERGEINIT; where the <i>outputfiles</i> parameter is an array; <i>outputfile</i> is an integer, specifying the MPE/3000 file number of the file on which the merged records are written. If <i>outputfile</i> is not specified, the records are not written anywhere. In this case, <i>postprocessor</i> must be specified.

All the other parameters are similar to the MERGEINIT parameters, except the positions of the *errorproc* and *keycompare* parameters are interchanged. MERGE is less powerful than MERGEINIT in that it does not have the *altseq* parameter. Also, MERGEEND and MERGEOUTPUT must not be called in this case. See Fig. 4-10 for an example of MERGE.

EXAMPLES

Calling MERGEINIT with the *preprocessor* parameter

```
$CONTROL USLINIT,FILE=21,FILE=22,FILE=23
  PROGRAM F13
  CHARACTER *72 BUF
  INTEGER KEYS(3),FNUM,OUTFILE(2)
  COMMON/FORTN/INFILE(3)
  INTEGER INFILE
  EXTERNAL PRE
  SYSTEM INTRINSIC MERGEINIT,MERGEEND
C
C  MERGE THE SORTED FILES, MAIL1(FTN21) AND MAIL2(FTN22), INTO
C  THE FILE, MAIL3(FTN23).
C  ESTABLISH THE MAJOR KEY AT 11 (LAST NAME) FOR 8 BYTES.
C  AND THE MINOR KEY AT 1 (FIRST NAME) FOR 9 BYTES.
  KEYS(1)=11
  KEYS(2)=8
  KEYS(3)=0
  KEYS(4)=1
  KEYS(5)=9
  KEYS(6)=0
C
C  ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
  INFILE(1)= FNUM(21)
  INFILE(2)=FNUM(22)
  INFILE(3)=0
  OUTFILE(1)=FNUM(23)
  OUTFILE(2)=0
C
C  INITIALIZE MERGE.
C
  CALL MERGEINIT(INFILE,PRE,OUTFILE,,,2,KEYS)
  CALL MERGEEND
C
C  READ AND DISPLAY THE OUTPUT FILE.
C
  REWIND 23
10 READ(23,END=100)BUF
  DISPLAY BUF
  GO TO 10
100 STOP
  END
```

PROGRAM UNIT F13 COMPILED

```

SUBROUTINE PRE(F1,CAR,N)
CHARACTER CAR(N)
INTEGER FNUM
COMMON/FORTN/INFILE(3)
INTEGER INFILE,F1
IF(INFILE(F1+1).EQ.FNUM(21))GO TO 1000
DO 90 J=61,72
CAR(J)="*"
90 CONTINUE
1000 RETURN
END

```

PROGRAM UNIT PRE COMPILED

```

:FILE FTN21=MAIL1, OLD
:FILE FTN22=MAIL2, OLD
:FILE FTN23=MAIL3, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

PLAINS	ANTELOPE	201 OPENSACE AVE	BIGCOUNTRY	WY	49301	369-732-4821
LOIS	ANYONE	6190 COURT ST	METROPOLIS	NY	20115	*****
KING	ARTHUR	329 EXCALIBUR ST	CAMELOT	CA	61322	*****
ALI	BABA	40 THIEVES WAY	SESAME	CO	69142	*****
BLACK	BEAR	47 ALLOVER DR	ANYWHERE	US	00111	NONE
JOHN	BIGTOWN	965 APPIAN WAY	METROPOLIS	NY	20013	*****
KNEE	BUCKLER	974 FISTICUFF DR	PUGILIST	ND	04321	*****
SWASH	BUCKLER	497 PLAYACTING CT	MOVIETOWN	CA	61497	*****
ANIMAL	CRACKERS	1000 ANYWHERE PL	ALLOVER	US	00001	001-100-1000
MULE	DEER	963 FOREST PL	NICECOUNTRY	CA	97643	493-900-9000
WHITETAIL	DEER	34 WOODSY PL	BACKCOUNTRY	ME	01341	619-433-4333
JAMES	DOE	4193 ANY ST	ANYTOWN	MD	00133	*****
JANE	DOE	3959 TREEWOOD LN	BIGTOWN	MA	21843	*****
PRAIRE	DOG	493 ROLLINGHILLS DR	OPENSACE	ND	24321	992-419-4192
JOHN	DOUGHE	239 MAIN ST	HOMETOWN	MA	26999	*****
MALLARD	DUCK	79 MARSH PL	PUDDLEDUCK	CA	97432	492-492-4922
JENNA	GRANDTR	493 TWENTIETH ST	PROGRESSIVE	CA	61335	*****
KARISSA	GRANDTR	7917 BROADMOOR WAY	BIGTOWN	MA	21799	*****
SNOWSHOE	HARE	742 FRIGID WAY	COLDSPOT	MN	37434	732-732-7320
MOUNTAIN	LION	796 KING DR	THICKET	NM	37643	712-712-7122
SPACE	MANN	9999 GALAXY WAY	UNIVERSE	CA	61239	*****
SWAMP	RABBIT	4444 DAMPLACE RD	BAYOU	LA	79999	NONE
NASTY	RATTLER	243 DANGER AVE	DESERTVILLE	CA	87654	828-432-4321
BIGHORN	SHEEP	999 MOUNTAIN DR	HIGHPLACE	CO	34567	776-409-9040
GREY	SQUIRREL	432 PLEASANT DR	FALLCOLORS	MA	14321	619-619-6199

Figure 4-2

INFILE and OUTFILE are the *inputfiles* and *outputfiles* parameters. The records in the files, MAIL1 and MAIL2, are compared on two keys. Note the *preprocessor*, PRE, replaces the telephone numbers from the file, MAIL2, by asterisks before MAIL1 and MAIL2 are merged into MAIL3.

Using the *keysonly* parameter



```
$CONTROL USLINIT,FILE=27,FILE=28,FILE=35
PROGRAM F14
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,INFILE(3),OUTFILE(2)
CHARACTER X
LOGICAL K
SYSTEM INTRINSIC MERGEINIT,MERGEEND

C
C ESTABLISH THE KEYS.
C MERGE THE FILES, AMERICAN AND REST, INTO FRIENDS.
C
    KEYS(1)=1
    KEYS(2)=17
    KEYS(3)=0

C
C ESTABLISH THE NUMBERS FOR THE FILES.
C
    INFILE(1)= FNUM(27)
    INFILE(2)=FNUM(28)
    INFILE(3)=0
    OUTFILE(1)=FNUM(35)
    OUTFILE(2)=0
    DISPLAY "SORT ON KEYS?"
    ACCEPT X
    IF(X.EQ."Y".OR.X.EQ."y")GO TO 18

C
C THE PARAMETER K IS TESTED FOR ITS TRUTH VALUE.
C
    K=.FALSE.
    GO TO 19
18 K=.TRUE.

C
C INITIALIZE MERGE WITH THE KEYSONLY PARAMETER, K.
C
19 CALL MERGEINIT(INFILE,,OUTFILE,,K,1,KEYS)
    CALL MERGEEND

C
C READ AND DISPLAY THE OUTPUT FILE.
C
    REWIND 35
10 READ(35,END=100)BUF
    DISPLAY BUF
    GO TO 10
100 STOP
    END
```

```
:FILE FTN27=AMERICAN, OLD
:FILE FTN28=REST, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

```
SORT ON KEYS? ?Y
```

```
Chamberlain,  
Chavez,  
Clift,  
Crane,  
Djilas,  
Hammarskjold,  
Horse,  
Joplin,  
K'ung,  
Khan,  
Lautreamont,  
Nijinsky,  
Noether,  
Ortega y Gasset,  
Pirandello,  
Rothstein,  
Sen,  
Truman,  
Vanderbilt,  
Wiener,
```

Figure 4-3

Only the key fields (last names) are sent as output since the *keyonly* parameter, K, is specified true.

```
:PREPRUN $OLDPASS; MAXDATA=15000
```

```
END OF PREPARE
```

```
SORT ON KEYS? ?N
```

Chamberlain,	Wilt	sportsman	born 1936
Chavez,	Cesar	labor leader	born 1927
Clift,	Montgomery	actor	born 1920
Crane,	Hart	poet	born 1899
Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
Horse,	Crazy	warrior	born 1848
Joplin,	Janis	singer	born 1943
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Rothstein,	Arnold	gangster	born 1882
Sen,	Mrinal	movie director	born 1923
Truman,	Harry	politician	born 1884
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894

Figure 4-4

The entire records are sent as output since the *keyonly* parameter, K, is specified false.

Calling MERGEINIT with the *statistics* parameter

```
$CONTROL USLINIT,FILE=27,FILE=28,FILE=35
PROGRAM F15
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,INFILE(3),OUTFILE(2),ST(10)
INTEGER NOINPUT, SPACE
INTEGER*4 NOREC, NOCOMP, CPUTME, ELTME
EQUIVALENCE (NOINPUT, ST(1)),(SPACE,ST(4))
EQUIVALENCE (NOREC, ST(2)), (NOCOMP, ST(5)), (CPUTME, ST(7))
EQUIVALENCE (ELTME,ST(9))
CHARACTER X
LOGICAL K
SYSTEM INTRINSIC MERGEINIT,MERGEEND

C
C MERGE THE FILES, AMERICAN AND REST, INTO THE FILE, WORLD.
C ESTABLISH THE KEYS.
C
KEYS(1)=1
KEYS(2)=17
KEYS(3)=0

C
C ESTABLISH THE NUMBERS FOR THE FILES.
C
INFILE(1)= FNUM(27)
INFILE(2)=FNUM(28)
INFILE(3)=0
OUTFILE(1)=FNUM(35)
OUTFILE(2)=0
DISPLAY "SORT ON KEYS?"
ACCEPT X
IF(X.EQ."Y".OR.X.EQ."y")GO TO 18

C
C THE PARAMETER, K, IS TESTED FOR ITS TRUTH VALUE.
C
K=.FALSE.
GO TO 19
18 K=.TRUE.

C
C INITIALIZE MERGE WITH THE KEYONLY PARAMETER, K.
C
19 CALL MERGEINIT(INFILE,,OUTFILE,,K,1,KEYS,,ST)
CALL MERGEEND
DISPLAY NOINPUT, NOREC, SPACE, NOCOMP, CPUTME, ELTME

C
C READ AND DISPLAY THE OUTPUT FILE.
C
REWIND 35
10 READ(35,END=100)BUF
DISPLAY BUF
GO TO 10
100 STOP
END
```



```

:FILE FTN27=AMERICAN, OLD
:FILE FTN28=REST, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

SORT ON KEYS? ?Y

	2	20	589	17	250	3608
Chamberlain,						
Chavez,						
Clift,						
Crane,						
Djilas,						
Hammarskjold,						
Horse,						
Joplin,						
K'ung,						
Khan,						
Lautreamont,						
Nijinsky,						
Noether,						
Ortega y Gasset,						
Pirandello,						
Rothstein,						
Sen,						
Truman,						
Vanderbilt,						
Wiener,						

Figure 4-5

The *statistics* parameter, ST, causes the MERGE statistics to be displayed.

Calling MERGEINIT with *charseq* parameter

```

$CONTROL USLINIT,FILE=31,FILE=32,FILE=33
PROGRAM F15
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,INFILE(3),OUTFILE(2),CSEQ(2)
SYSTEM INTRINSIC MERGEINIT,MERGEEND
C
C MERGE THE FILES, SWED1 (FTN31), SWED2 (FTN32), INTO THE
C FILE SWED3 (FTN33), USING THE COLLATING SEQUENCE FOR THE
C NATIVE LANGUAGE, SWEDISH.
C
C ESTABLISH THE KEY.
C
KEYS(1)=1
KEYS(2)=10
KEYS(3)=9
C
C ESTABLISH THE CHARSEQ PARAMETER; LANGUAGE = SWEDISH
C LANGUAGE ID = 13
C

```

```

      CSEQ(1)=1
      CSEQ(2)=13
C
C   ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES
C
      INFILE(1)=FNUM(31)
      INFILE(2)=FNUM(32)
      INFILE(3)=0
      OUTFILE(1)=FNUM(33)
      OUTFILE(2)=0
C
C   INITIALIZE MERGE USING THE charseq PARAMETER
C
      CALL MERGEINIT(INFILE,,OUTFILE,,,1,KEYS,,,,,,CSEQ)
      CALL MERGEEND
C
C   READ AND DISPLAY THE OUTPUT FILE
C
      REWIND 33
10    READ(33,END=100) BUF
      DISPLAY BUF
      GO TO 10
100   STOP
      END

```

PROGRAM UNIT F15 COMPILED

```

****      GLOBAL STATISTICS      ****
**** NO ERRORS,  NO WARNINGS ****
TOTAL COMPILATION TIME 0:00:01
TOTAL ELAPSED TIME    0:00:02
TOTAL SYMBOL TABLE USED  % 07

```

END OF COMPILE

```

END OF PREPARE
:FILE FTN31=SWED1,old
:FILE FTN32=SWED2,old
:FILE FTN33=SWED3,new
:PREPRUN $OLDPASS;MAXDATA=32000

```

```

Anderson, Hans
Bergman, Ingrid
Bergman, Ingmar
Borg, Bjorn
Hammerskjold, Dag
Sverson, Bo
Ullman, Liv

```

Figure 4-5a

Calling MERGEOUTPUT

```

$CONTROL USLINIT,FILE=27,FILE=28,FILE=35
PROGRAM F16
CHARACTER*72 BUF
LOGICAL LBUF(36)
EQUIVALENCE (LBUF,BUF)
INTEGER KEYS(3),FNUM,INFILE(3),LENGTH
SYSTEM INTRINSIC MERGEINIT,MERGEEND,MERGEOUTPUT

```

C


```

C   MERGE THE FILES, AMERICAN AND REST, INTO THE FILE, FRIENDS.
C   ESTABLISH THE KEYS.
C
      KEYS(1)=1
      KEYS(2)=15
      KEYS(3)=0
C
C   ESTABLISH THE NUMBERS FOR THE FILES.
C
      INFILE(1)= FNUM(27)
      INFILE(2)=FNUM(28)
      INFILE(3)=0
C
C   INITIALIZE MERGE WITHOUT THE OUTPUTFILES PARAMETER.
C
      CALL MERGEINIT(INFILE,,,,,1,KEYS)
5   CALL MERGEDOUTPUT(LBUF,LENGTH)
      IF (LENGTH .LE. -1) GOTO 9
      DISPLAY BUF
      GOTO 5
9   CALL MERGEEND
100 STOP
      END

```

```

:FILE FTN27=AMERICAN, OLD
:FILE FTN28=REST, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=4000

```

END OF PREPARE

Chamberlain,	Wilt	sportsman	born 1936
Chavez,	Cesar	labor leader	born 1927
Clift,	Montgomery	actor	born 1920
Crane,	Hart	poet	born 1899
Djilas,	Milovan	sociologist	born 1911
Hammar skjold,	Dag	pacifist	born 1905
Horse,	Crazy	warrior	born 1848
Joplin,	Janis	singer	born 1943
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Rothstein,	Arnold	gangster	born 1882
Sen,	Mrinal	movie director	born 1923
Truman,	Harry	politician	born 1884
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894

Figure 4-6

Note the *outputfiles* parameter is not specified in this case.

Calling MERGESTAT

```

$CONTROL USLIMIT,FILE=27,FILE=28,FILE=35
  PROGRAM F17
  CHARACTER *72 BUF
  INTEGER KEYS(3),FNUM,INFILE(3),OUTFILE(2),ST(10)
  CHARACTER X
  LOGICAL K
  SYSTEM INTRINSIC MERGEINIT,MERGEEND,MERGESTAT
C
C MERGE THE FILES, AMERICAN AND REST, INTO THE FILE, WORLD.
C ESTABLISH THE KEYS.
C
  KEYS(1)=1
  KEYS(2)=17
  KEYS(3)=0
C
C ESTABLISH THE NUMBERS FOR THE FILES.
C
  INFILE(1)= FNUM(27)
  INFILE(2)=FNUM(28)
  INFILE(3)=0
  OUTFILE(1)=FNUM(35)
  OUTFILE(2)=0
  DISPLAY "SORT ON KEYS?"
  ACCEPT X
  IF(X.EQ."Y".OR.X.EQ."y")GO TO 18
C
C THE PARAMETER, K, IS TESTED FOR ITS TRUTH VALUE.
C
  K=.FALSE.
  GO TO 19
18 K=.TRUE.
C
C INITIALIZE MERGE WITH THE KEYSONLY PARAMETER, K.
C
19 CALL MERGEINIT(INFILE,,OUTFILE,,K,1,KEYS,,ST)
  CALL MERGEEND
C
C CALL THE MERGESTAT INTRINSIC
C
  CALL MERGESTAT(ST)
C
C READ AND DISPLAY THE OUTPUT FILE.
C
  REWIND 35
10 READ(35,END=100)BUF
  DISPLAY BUF
  GO TO 10
100 STOP
  END
```

```
:FILE FTN27=AMERICAN, OLD
:FILE FTN28=REST, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

SORT ON KEYS? ?Y



STATISTICS

```
NUMBER OF INPUT FILES =
NUMBER OF RECORDS =
SPACE AVAILABLE (IN WORDS) =
NUMBER OF COMPARES =
CPU TIME (MINUTES) =
ELAPSED TIME (MINUTES) =
```

```
Chamberlain,
Chavez,
Clift,
Crane,
Djilas,
Hammaraskjold,
Horse,
Joplin,
K'ung,
Khan,
Lautreamont,
Nijinsky,
Noether,
Ortega y Gasset,
Pirandello,
Rothstein,
Sen,
Truman,
Vanderbilt,
Wiener,
```

Figure 4-7

Calling MERGETITLE

```
$CONTROL USLINIT,FILE=27,FILE=28,FILE=35
PROGRAM F18
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,INFILE(3),OUTFILE(2)
SYSTEM INTRINSIC MERGEINIT,MERGEEND,MERGETITLE
```

C

```

C   MERGE THE FILES, AMERICAN AND REST, INTO THE FILE, WORLD.
C   ESTABLISH THE KEYS.
C
      KEYS(1)=1
      KEYS(2)=15
      KEYS(3)=0
      INFILE(1)= FNUM(27)
      INFILE(2)=FNUM(28)
      INFILE(3)=0
      OUTFILE(1)=FNUM(35)
      OUTFILE(2)=0
C
C   INITIALIZE MERGE.
C
      CALL MERGEINIT(INFILE,,OUTFILE,,,1,KEYS)
C
C   CALL THE MERGETITLE INTRINSIC.
C
      CALL MERGETITLE
      CALL MERGEEND
C
C   READ AND DISPLAY THE OUTPUT FILE.
C
      REWIND 35
      10 READ(35,END=100)BUF
         DISPLAY BUF
         GO TO 10
      100 STOP
         END

```

```

:FILE FTN27=AMERICAN, OLD
:FILE FTN28=REST, OLD
:FILE FTN35=WORLD, NEW
:PREPRUN $OLDPASS; MAXDATA=4000

```

END OF PREPARE

Chamberlain,	Wilt	sportsman	born 1936
Chavez,	Cesar	labor leader	born 1927
Clift,	Montgomery	actor	born 1920
Crane,	Hart	poet	born 1899
Djilas,	Milovan	sociologist	born 1911
Hammarskjold,	Dag	pacifist	born 1905
Horse,	Crazy	warrior	born 1848
Joplin,	Janis	singer	born 1943
K'ung,	Ch'iu	preacher	born 551 B.C.
Khan,	Jenghiz	emperor	born 1167 (?)
Lautreamont,	Comte de	novelist	born 1846
Nijinsky,	Vaslav	dancer	born 1890
Noether,	Emmy	mathematician	born 1882
Ortega y Gasset,	Jose	philosopher	born 1883
Pirandello,	Luigi	playright	born 1867
Rothstein,	Arnold	gangster	born 1882
Sen,	Mrinal	movie director	born 1923
Truman,	Harry	politician	born 1884
Vanderbilt,	Cornelius	capitalist	born 1794
Wiener,	Norbert	cybernetician	born 1894

Figure 4-8

Calling MERGEERRORMESS from the *errorproc* subroutine ER

```
$CONTROL USLINIT,FILE=21,FILE=22,FILE=23
PROGRAM F19
CHARACTER *72 BUF
INTEGER KEYS(3),FNUM,OUTFILE(2)
COMMON/FORTN/INFILE(3)
INTEGER INFILE
EXTERNAL PRE,ER
SYSTEM INTRINSIC MERGEINIT,MERGEEND

C
C MERGE SORTED FILES, MAIL1 (FTN21) AND MAIL2 (FTN22), INTO
C THE FILE, MAIL3 (FTN23).
C ESTABLISH THE MAJOR KEY AT 0 FOR 8 BYTES.
C AND THE MINOR KEY AT 0 FOR 9 BYTES.
KEYS(1)=0
KEYS(2)=8
KEYS(3)=0
KEYS(4)=0
KEYS(5)=9
KEYS(6)=0

C
```



```

C ESTABLISH NUMBERS FOR THE INPUT AND OUTPUT FILES.
C
    INFILE(1)= FNUM(21)
    INFILE(2)=FNUM(22)
    INFILE(3)=0
    OUTFILE(1)=FNUM(23)
    OUTFILE(2)=0
C
C INITIALIZE MERGE.
C
    CALL MERGEINIT(INFILE,PRE,OUTFILE,,,2,KEYS,,,ER)
    CALL MERGEEND
C
C READ AND DISPLAY THE OUTPUT FILE.
C
    REWIND 23
10 READ(23,END=100)BUF
    DISPLAY BUF
    GO TO 10
100 STOP
    END

```

PROGRAM UNIT F19 COMPILED

```

    SUBROUTINE PRE(F1,CAR,N)
    CHARACTER CAR(N)
    INTEGER FNUM
    COMMON/FORTN/INFILE(3)
    INTEGER INFILE,F1
    IF(INFILE(F1+1).EQ.FNUM(21))GO TO 1000
    DO 90 J=61,72
    CAR(J)="*"
    90 CONTINUE
1000 RETURN
    END

```

PROGRAM UNIT PRE COMPILED

```

    SUBROUTINE ER(ERR)
    INTEGER ERR
    CHARACTER *72 MESSAGE
    SYSTEM INTRINSIC MERGEERRORMESS
    MESSAGE=""
    CALL MERGEERRORMESS(ERR,MESSAGE,L)
    DISPLAY MESSAGE
    RETURN
    END

```

PROGRAM UNIT ER COMPILED

```

:FILE FTN21=MAIL1, OLD
:FILE FTN22=MAIL2, OLD
:FILE FTN23=MAIL3, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

```

MERGELIB: KEYFIELD IS NOT WITHIN RECORD LENGTH OF EACH FILE
ABORT :$OLDPASS.PUB.JOSHI.?.?:GRSL.%1.%3
PROGRAM ERROR #24 :BOUNDS VIOLATION

PROGRAM TERMINATED IN AN ERROR STATE. (CIERR 976)

```

Figure 4-9

Note the keys are established at the column position zero.

Calling MERGE with the *failure* parameter

```

$CONTROL USLINIT,FILE=21,FILE=22,FILE=23
PROGRAM F20
CHARACTER BUF*72
INTEGER KEYS(6),FNUM,INFILES(2)
LOGICAL FAILURE

C
C MERGE TWO SORTED FILES, MAIL1 (FTN21) AND MAIL2 (FTN22),
C INTO A THIRD FILE, MAIL3 (FTN23)
C
C ESTABLISH KEYS - MAJOR AT 11 FOR 9 BYTES (LAST NAME)
C MINOR AT 1 FOR 10 BYTES (FIRST NAME)
C
      KEYS(1)=11
      KEYS(2)=9
      KEYS(3)=0
      KEYS(4)=1
      KEYS(5)=10
      KEYS(6)=0

C
C ESTABLISH NUMBERS FOR THE INPUT FILES, MAIL1 AND MAIL2.
C
      INFILES(1)=FNUM(21)
      INFILES(2)=FNUM(22)

C
C INITIALIZE MERGE - 2 KEYS ARE SPECIFIED
C
      CALL MERGE(\2\,INFILES,\FNUM(23)\,\0\,\2\,KEYS,
# \0\,\0\,\0\,\0\,\0\,FAILURE,\%7301\
      IF(FAILURE)STOP 10

C
C READ AND DISPLAY OUTPUT FILE
C
      REWIND 23
20      READ(23,END=30)BUF
          DISPLAY BUF[1:72]
          GO TO 20
30      STOP
          END

```

```

:FILE FTN21=MAIL1, OLD
:FILE FTN22=MAIL2, OLD
:FILE FTN23=TEST, NEW
:PREPRUN $OLDPASS; MAXDATA=15000

```

END OF PREPARE

PLAINS	ANTELOPE	201 OPENSACE AVE	BIGCOUNTRY	WY 49301	369-732-4821
LOIS	ANYONE	6190 COURT ST	METROPOLIS	NY 20115	619-732-4997
KING	ARTHUR	329 EXCALIBUR ST	CAMELOT	CA 61322	812-200-0100
ALI	BABA	40 THIEVES WAY	SESAME	CO 69142	NONE
BLACK	BEAR	47 ALLOVER DR	ANYWHERE	US 00111	NONE
JOHN	BIGTOWN	965 APPIAN WAY	METROPOLIS	NY 20013	619-407-2314
KNEE	BUCKLER	974 FISTICUFF DR	PUGILIST	ND 04321	976-299-2990
SWASH	BUCKLER	497 PLAYACTING CT	MOVIETOWN	CA 61497	NONE
ANIMAL	CRACKERS	1000 ANYWHERE PL	ALLOVER	US 00001	001-100-1000
MULE	DEER	963 FOREST PL	NICECOUNTRY	CA 97643	493-900-9000
WHITETAIL	DEER	34 WOODSY PL	BACKCOUNTRY	ME 01341	619-433-4333
JAMES	DOE	4193 ANY ST	ANYTOWN	MD 00133	237-408-7100
JANE	DOE	3959 TREEWOOD LN	BIGTOWN	MA 21843	714-399-4563
PRAIRE	DOG	493 ROLLINGHILLS DR	OPENSACE	ND 24321	992-419-4192
JOHN	DOUGHE	239 MAIN ST	HOMETOWN	MA 26999	714-411-1123
MALLARD	DUCK	79 MARSH PL	PUDDLEDUCK	CA 97432	492-492-4922
JENNA	GRANDTR	493 TWENTIETH ST	PROGRESSIVE	CA 61335	799-191-9191
KARISSA	GRANDTR	7917 BROADMOOR WAY	BIGTOWN	MA 21799	713-244-3717
SNOWSHOE	HARE	742 FRIGID WAY	COLDSPOT	MN 37434	732-732-7320
MOUNTAIN	LION	796 KING DR	THICKET	NM 37643	712-712-7122
SPACE	MANN	9999 GALAXY WAY	UNIVERSE	CA 61239	231-999-9999
SWAMP	RABBIT	4444 DAMPPLACE RD	BAYOU	LA 79999	NONE
NASTY	RATTLER	243 DANGER AVE	DESERTVILLE	CA 87654	828-432-4321
BIGHORN	SHEEP	999 MOUNTAIN DR	HIGHPLACE	CO 34567	776-409-9040
GREY	SQUIRREL	432 PLEASANT DR	FALLCOLORS	MA 14321	619-619-6199

Figure 4-10

The files, MAIL1 and MAIL2, are merged into the file, MAIL3.

Calling MERGE for Multi-record, NOBUF, and Buffered Files

:FORTRAN NOBUF2

PAGE 0001 HP32102B.01.02 FORTRAN/3000 (C) HEWLETT-PACKARD CO. 1979 TUE, JAN

```
$CONTROL FILE=10-13,USLINIT
PROGRAM PNOBUF2

C
C This program demonstrates the use of multirecord, NOBUF,
C and buffered files with the MERGE intrinsics. The failure
C parameter is checked and the errorparm is used to obtain
C the error message text if needed.
C
C
C SYSTEM INTRINSIC MERGEINIT,MERGEEND,MERGESTAT,MERGEERRORMESS
C INTEGER INPUT(4),OUTPUT(2),KEYS(3),STATS(12)
C INTEGER ERROR,FNUM,MSGLEN
C LOGICAL FAILED
C CHARACTER*72 MSG

C
C Establish the file numbers for the input and output files
C
C INPUT(1) = FNUM(11)
C INPUT(2) = FNUM(12)
C INPUT(3) = FNUM(13)
C INPUT(4) = 0

C
C OUTPUT(1) = FNUM(10)
C OUTPUT(2) = 0

C
C Keys to sort the sequence numbers in columns 73-80
C
C KEYS(1) = 73
C KEYS(2) = 8
C KEYS(3) = 0

C
C FAILED = .FALSE.
C ERROR = 0

C
C Sort the files.
C
C CALL MERGEINIT(INPUT,,OUTPUT,,,1,KEYS,,,STATS,FAILED,ERROR)
C CALL MERGEEND

C
C Print error message if one occurred, otherwise statistics.
C
C IF (FAILED) GO TO 100

C
C CALL MERGESTAT(STATS)
C GOTO 200
100 CALL MERGEERRORMESS(ERROR,MSG,MSGLEN)
C DISPLAY MSG[1:MSGLEN],` ` ( ` `,ERROR,` ` )` `
200 STOP
END
```

PROGRAM UNIT PNOBUF2 COMPILED

**** GLOBAL STATISTICS ****
**** NO ERRORS, NO WARNINGS ****
TOTAL COMPILATION TIME 0:00:02
TOTAL ELAPSED TIME 0:00:04

END OF COMPILE
:PREP \$OLDPASS,PNOBUF2;MAXDATA=31232

END OF PREPARE
:COMMENT INPUT FILES
:FILE FTN11=SORTED01,OLD;MR
:FILE FTN12=SORTED02,OLD;NOBUF
:FILE FTN13=SORTED03,OLD
:COMMENT OUTPUT FILE
:FILE FTN10=MERGED,NEW;MR;SAVE
:RUN PNOBUF2;LIB=G

STATISTICS

NUMBER OF INPUT FILES =	3
NUMBER OF RECORDS =	150
SPACE AVAILABLE (IN WORDS) =	28,160
NUMBER OF COMPARES =	243
CPU TIME (MINUTES) =	.01
ELAPSED TIME (MINUTES) =	.04

END OF PROGRAM

CALLING SORT FROM A SPL/3000 PROGRAM

SECTION

V

This section introduces you to the programmatic use of SORT-MERGE/3000. You can sort one or more files from an SPL/3000 program by using intrinsic calls. These intrinsics (SPL/3000 procedures) are part of SORT-MERGE/3000 and are called by using the SYSTEM INTRINSIC declarations in your program. The various parameters of these intrinsics are used by SORT-MERGE/3000 to perform specific operations.



The SORT program intrinsics

The following is a list of the SORT program intrinsics which reside in the SORTLIB segment of the system segmented library (SL.PUB.SYS):

INTRINSIC	DESCRIPTION
SORTINIT	Initiates the SORT operation.
SORTINPUT	Passes the input records, one at a time, to the SORT program only if the <i>inputfiles</i> parameter is not specified in SORTINIT.
SORTOUTPUT	Signals the beginning of SORT and receives each output record from SORT into an array specified by the <i>record</i> parameter. SORTOUTPUT signals the end of the input process if SORTINPUT is also called. SORTOUTPUT is used only if the <i>outputfiles</i> parameter of SORTINIT is not specified.
SORTEND	Closes the scratch file and restores the data stack to its original state. It signals the beginning of SORT if SORTOUTPUT is not called.
SORTSTAT	Prints the SORT statistics on \$STDLIST.
SORTTITLE	Prints the version number and title of the SORTLIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.
SORTERRORMESS	Called to retrieve and print a message if a fatal error occurs during SORT. SORTERRORMESS is called from a user supplied error procedure (the <i>errorproc</i> parameter of SORTINIT).

The call to SORTINIT starts the SORT operation. You should follow it by calling SORTINPUT if the *inputfiles* parameter of SORTINIT is not specified. After this, call SORTOUTPUT if the *outputfiles* parameter of SORTINIT is not specified. Then call SORTEND to terminate SORT. SORTINIT and SORTEND must be called from the same procedure. If you want the display of the SORT statistics, call SORTSTAT. Additionally, call SORTERRORMESS from the user supplied procedure, *errorproc*, if you want a display of the message when an error occurs. Note SORTSTAT, and SORTERRORMESS are optional. However, their order is important whenever they are called. Optional intrinsic SORTTITLE is an exception in that it can be called from the program at any stage after the declaration of the system intrinsics. The following flowchart describes the SORT operation when SORTINPUT, SORTOUTPUT, and SORTSTAT are used:

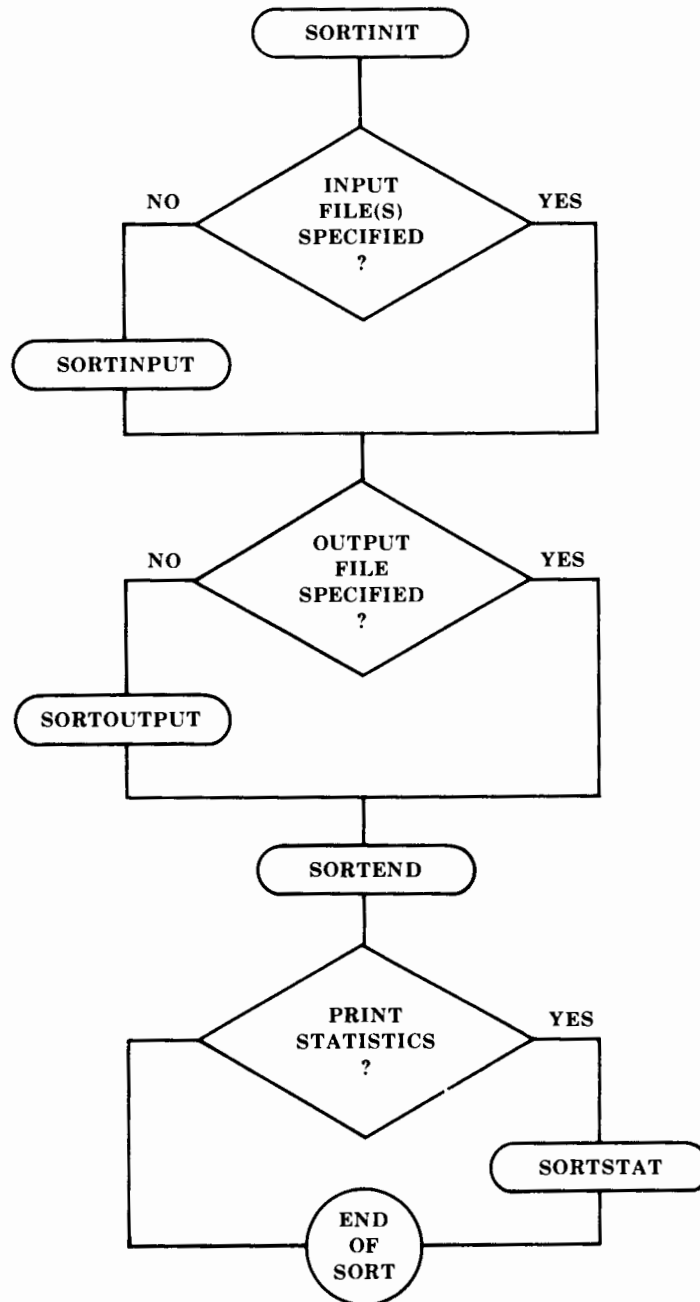


Figure 5-1. Flowchart of SORT Intrinsic

Preparation and Execution of the SORT Programs

The amount of available memory can affect both the time required to perform the SORT operation and the amount of secondary storage needed by a temporary file. SORT programs should normally be prepared with the maximum available segsize, which is specified by the MAXDATA= segsize parameter of the :PREP or :RUN commands.

If a smaller segsize must be used, the following allowances should be made to provide enough space for sorting. The segsize should be approximately 12000 words greater than the space required to run your program without calls to the SORT intrinsics. If the files to be sorted are opened NOBUF, the segsize should be increased by an additional amount equal to your blocksize (in words). When sorting files that have been opened multirecord, the maximum possible segsize should be used.

If the error message INSUFFICIENT STACK SPACE is displayed, increase the MAXDATA parameter. If the message TOO MANY FILES OPEN (FSERR 71) appears, it means MPE has no room for its tables in the user data segment. Use the NOCB parameter of the :RUN command during the execution of the program in this case.

NOTE: The SORTINITIAL intrinsic is included in this manual for the maintenance of existing SPL/3000 programs.

SORTINIT

Initiates the SORT operation.

SYNTAX

```
PROCEDURE SORTINIT ( IAinputfiles, IAoutputfiles, IVoutputoption,  
IVreclen, DVnumrecs, IVnumkeys, IAkeys, IAaltseq,  
LPkeycompare, Perrorproc, IAstatistics, Lfailure,  
Ierrorparm, Ispaceallocation, IAcharseq, parm1); O-V
```

PARAMETERS

- inputfiles* An integer array containing the MPE/3000 file identification numbers of the files to be sorted. The array must be terminated with a word of zero to indicate the end of the list. If the files are opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/ deblocking. \$Null is not a valid input file.
- outputfiles* An integer array containing the file identification number of the output file. The second word must contain a zero to indicate the end of the list. If the file is opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking.
- outputoption* An integer passed by value determining the format of the output records. There are four possibilities:
- 0 - Output record is the same as the input record (default value)
 - 1 - Output record is a double integer (4 characters) whose value is the logical (relative) record number of the record.
 - 2 - Output record contains only the key fields, concatenated together with the major keys on the left followed by the remaining keys.
 - 3 - Output record is the logical record number followed by the key fields.
- reclen* An integer passed by value denoting the maximum length of a record in characters. If it is not specified, the record length is taken from the first file specified in the *inputfiles* array. In this case, you must specify the *inputfiles* parameter.
- numrecs* A double integer passed by value denoting the upper bound to the number of records sorted. If this is not specified (or if all the input files are not on the disc), the value of 10,000 (double integer) is used. Otherwise, the parameter value is derived from the file label (the end of the file number of the input files).

numkeys and *keys*

numkeys is an integer passed by value and *keys* is an integer array. They specify the way the records are sorted. If either is specified, the other must also be specified and the *keycompare* parameter must not be specified. *numkeys* is the number of keys used in the comparison of records and must be either equal to or greater than one. For each key being specified, *keys* contains three words.

First word gives the position of the first character of the key within the input record. (The first character of the record is considered to be in position 1.) Second word denotes the total number of characters in the key. Third word (bits 0 through 7) gives the ordering sequence of the records; 0 for ascending, 1 for descending. bits 8 through 15 of the third word indicate the type of data according to the following convention:

0=logical or byte (same as the *type*, BYTE, in interactive mode)

1=two's complement (including integer and double integer)

2=floating point (including real and long)

3=packed decimal

5=packed decimal with even number of digits

4=Display-Trailing-Sign (see the KEY command in Section II)

6=Display-Leading-Sign

7=Display-Leading-Sign-Separate

8= Display-Trailing-Sign-Separate

9= Character (Collating sequence of *charseq* [see below] is used)

Note: The integrity of the integer array *keys* must be maintained throughout the sorting operation. Do not change the array until after a call has been made to the intrinsic SORTEND.

altseq

An integer array defining an alternate collating sequence. The first character (bits 0-7) of the array is defined according to Table 3-1. The second character (bits 8-15) specifies one less than the total number of characters in the collating sequence (in this case, 255 or %377). These two characters are followed by the actual collating sequence responsible for the particular SORT operation. See Fig. 5-1 for details.

keycompare

A logical procedure allowing you access to your records when they are compared. It must be specified in your call to SORTINIT if you do not specify *numkeys* and *keys*. This logical procedure should include a statement of the following form:

```
LOGICAL PROCEDURE keycompare (rec1, len1, rec2, len2);
```

rec1 and *rec2* are byte arrays and are pointers to the two records. *len1* and *len2* are integers passed by reference and are the lengths of the records in characters. *keycompare* returns a true value if *rec1* precedes *rec2*, and a false value otherwise. A true value is also returned in the case of ties, to ensure that the records with equal keys retain their original order.

errorproc A procedure called whenever a fatal error occurs during SORT. It is called by a statement of the following form:

```
PROCEDURE errorproc (errorcode);
```

errorcode is an integer passed by reference and is the SORT program error number. It is passed to *errorproc* when an error occurs. If *errorproc* or *errorparm* are not specified, a default procedure is used which prints an error message corresponding to the particular *errorcode*. For a list of these error messages, see Appendix A.

■ *statistics* An integer array, which if specified, gives the following data:

Zeroth and first words=
number of records sorted (double integer)

Second word=
number of intermediate passes

Third word=
space available for sorting

Fourth and fifth words=
number of comparisons (double integer)

Sixth and seventh words=
number of scratch file inputs/outputs (double integer)

Eighth and ninth words=
CPU time used (in milliseconds, double integer)

Tenth and eleventh words=
elapsed time (in milliseconds, double integer)

failure A logical word passed by reference which is set to -1 (true) if a fatal error occurs, and 0 (false) otherwise. It is set after each call to SORTINPUT and SORTOUTPUT; in addition, the condition code is set.

Error conditions:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

<i>errorparm</i>	An integer variable which, if specified, is set to the SORTLIB error number if an error occurs. The SORTERRORMESS intrinsic can be used to obtain the error message text. If the errorparm is supplied, the errorproc procedure is ignored and no error messages are display. For a list of error messages see Appendix A.
<i>spaceallocation</i>	An integer variable which, if specified, is used to determine stack allocation. A positive spaceallocation specifies the number of words that may be used for sorting and buffering. A negative values specifies the number of words that should be left for the user after determining the amount available. Zero will cause a default value to be used.
<i>charseq</i>	2 word integer array. Set word 0 to 1. Set word 1 to <i>langnum</i> of the native language whose collating sequence is to be used to sort keys of type 9 (character).
<i>parm1</i>	unused.

SORTINPUT

Passes the input records, one at a time, to the SORT program, only if the *inputfiles* parameter is not specified in SORTINIT.

SYNTAX

```
PROCEDURE SORTINPUT ( LArecord, IVlength);
```

PARAMETERS

<i>record</i>	A logical array containing a data record.
<i>length</i>	An integer passed by value denoting the number of characters in the record. It must be long enough to contain all the keys, but not longer than the record size (<i>reclen</i>).

ERROR CONDITIONS:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

This intrinsic follows SORTINIT and precedes SORTOUTPUT and SORTEND (see Fig. 5-2).

SORTOUTPUT

Signals the beginning of SORT and receives each output record from SORT into an array specified by the *record* parameter. SORTOUTPUT signals the end of the input process if SORTINPUT is also called. SORTOUTPUT is called only if the *outputfiles* parameter of SORTINIT is not specified (Fig. 5-3).

SYNTAX

```
PROCEDURE SORTOUTPUT ( LA record, I length );
```

PARAMETERS

record A logical array receiving the next output record in the format specified by *outputoption*.

length An integer passed by reference denoting the number of characters returned in the record. When no more records remain, *length* is set to -1.

ERROR CONDITIONS:

CCE=
no error occurred (*failure* set to false)

CCL=
error occurred (*failure* set to true)

Note if SORTINPUT is also called, SORTOUTPUT is called only after SORTINPUT has passed all the records (Fig. 5-4). SORTOUTPUT always precedes SORTEND.

SORTEND

Closes the scratch file and restores the data stack to its original state. It signals the beginning of SORT if SORTOUTPUT is not called.

SYNTAX

PROCEDURE SORTEND;

ERROR CONDITIONS:

CCE=

no error occurred during SORT (*failure set to false*)

CCL=

an error occurred during SORT (*failure set to true*)

SORTEND is required if SORTINIT is called. It can be called either after all the calls to the output file are completed by SORTINIT, or after all the calls to SORTOUTPUT are completed.

- Note: SORTEND must be called from the same subroutine that called SORTINIT.

SORTSTAT

Prints the SORT statistics on \$STDLIST.

SYNTAX

```
PROCEDURE SORTSTAT ( IAstatistics);
```

statistics is an integer array. SORTSTAT is called after SORTEND.

SORTTITLE

Prints the version number and title of the SORTLIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.

SYNTAX

PROCEDURE SORTTITLE;

This intrinsic can be called from the program at any stage after the declaration of the system intrinsics.

SORTERRORMESS

Called to retrieve and print a message if a fatal error occurs during SORT. SORTERRORMESS is called from a user supplied procedure (the *errorproc* parameter of SORTINIT).

SYNTAX

```
PROCEDURE SORTERRORMESS ( IVerrorcode, BAmessage, Ilength);
```

PARAMETERS

<i>errorcode</i>	An integer passed by value denoting the SORT program error number and is passed to <i>errorproc</i> when an error occurs.
<i>message</i>	A byte array containing the text of the message. The <i>message</i> parameter must be at least 72 characters long.
<i>length</i>	An integer passed by reference denoting the length of the message in characters.

SORTERRORMESS works in conjunction with the *errorproc* parameter of SORTINIT.

SORTINITIAL

Initiates the SORT operation (to be used only for existing SPL/3000 programs).

SYNTAX

```
PROCEDURE SORTINITIAL ( IV inputfile, IV outputfile, IV outputoption,  
                       IV reclen, DV numrecs, IV numkeys, LA keys, P errorproc,  
                       LP keycompare, LA statistics, L failure);
```

PARAMETERS

- | | |
|-------------------|--|
| <i>inputfile</i> | An integer passed by value which is the MPE/3000 file number of the file to be sorted. Input records are read directly from the file by the SORT program, and no calls are made to SORTINPUT. If <i>inputfile</i> is not specified, the records are passed via SORTINPUT which must be called. |
| <i>outputfile</i> | An integer passed by value which is the MPE/3000 file number of the file receiving the sorted records. If specified, no calls to SORTOUTPUT may be made. Otherwise, the sorted records are sent through the SORTOUTPUT procedure, which must be called. |

Unlike SORTINIT, where the *inputfiles* and *outputfiles* parameters are integer arrays, the parameters, *inputfile* and *outputfile*, are integers passed by value; each of them representing only a single file. Parameters, *keys* and *statistics* are logical arrays. SORTINITIAL does not have the capability of defining an alternate collating sequence. Also, the positions of the *errorproc* and *keycompare* parameters are interchanged. The remaining parameters follow the same rules as those in SORTINIT.

EXAMPLES

Calling SORTINIT with the *altseq* Parameter

```
00000 0 $CONTROL USLINIT
00000 0 << SPL EXAMPLE S1 >>
00000 0 <<SPL ALTSEQ PARAMETER EXAMPLE>>
00000 0 <<INPUTFILE AND OUTPUTFILE SPECIFIED>>
00000 0 <<SORT THE FILE, UNDRGRAD, INTO THE FILE, VICTORS>>
00000 0 <<SORT ON GRADES>>
00000 0 BEGIN
00000 1 BYTE ARRAY INPUT(0:12):=""UNDRGRAD  ";
00010 1 BYTE ARRAY OUTPUT(0:9):=""VICTORS  ";
00006 1 ARRAY BUF(0:21);
00006 1 INTEGER ARRAY IN(0:1),OUT(0:1);
00006 1 INTEGER LEN;
00006 1 INTEGER NUMKEYS:=1;
00006 1 INTEGER ARRAY KEYS(0:10):=38,1,0;
00003 1 INTEGER ARRAY ALTSEQ(0:130):=
00000 1 %000377,
00001 1 %000001,%001003,%002005,%003007,%004011,%005013,%006015,%007017,
00011 1 %010021,%011023,%012025,%013027,%014031,%015033,%016035,%017037,
00021 1 %020041,%021043,%022045,%023047,%024051,%025053,%026055,%027057,
00031 1 %030061,%031063,%032065,%033067,%034071,%035073,%036075,%037077,
00041 1 %040102,%040503,%042105,%043107,%044111,%045113,%046115,%047117,
00051 1 %050121,%051123,%052125,%053127,%054131,%055133,%056135,%057137,
00061 1 %060141,%061143,%062145,%063147,%064151,%065153,%066155,%067157,
00071 1 %070161,%071163,%072165,%073167,%074171,%075173,%076175,%077177,
00101 1 %100201,%101203,%102205,%103207,%104211,%105213,%106215,%107217,
00111 1 %110221,%111223,%112225,%113227,%114231,%115233,%116235,%117237,
00121 1 %120241,%121243,%122245,%123247,%124251,%125253,%126255,%127257,
00131 1 %130261,%131263,%132265,%133267,%134271,%135273,%136275,%137277,
00141 1 %140301,%141303,%142305,%143307,%144311,%145313,%146315,%147317,
00151 1 %150321,%151323,%152325,%153327,%154331,%155333,%156335,%157337,
00161 1 %160341,%161343,%162345,%167747,%164351,%165353,%166355,%167357,
00171 1 %170361,%171363,%172365,%173367,%174371,%175373,%176375,%177377;
00201 1 INTRINSIC FPOINT,QUIT,FOPEN,FREAD,PRINT,SORTINIT,SORTEND;
00201 1 IN(0):=FOPEN(INPUT,5);
00010 1 IN(1):=0;
00013 1 IF <> THEN QUIT(90);
00016 1 OUT(0):=FOPEN(OUTPUT,4,4);
00027 1 OUT(1):=0;
00032 1 IF <> THEN QUIT(91);
00035 1 <<CALL SORTINIT -OUTPUT OPTION=0>>
00035 1 SORTINIT(IN,OUT,,,,NUMKEYS,KEYS,ALTSEQ);
00046 1 SORTEND;
00047 1 FPOINT(OUT,0D);
00052 1 LOOP: LEN:=FREAD(OUT,BUF,21);
00060 1 IF <> THEN QUIT(93);
00063 1 PRINT(BUF,LEN,0);
00067 1 GOTO LOOP;
00073 1 END.
```

```
:BUILD VICTORS
:PREPRUN $OLDPASS; MAXDATA=15000
```

```
END OF PREPARE
```

Nicolas Bourbaki	4.0	B
Sensible Kommunist	3.6	B
Milind Ranade	3.9	B
Uncle Samuelson	3.7	B
Vegetarian Dracula	3.8	B
Virgin Cat	3.1	A
Tech Nitpicker	3.2	A
Boris Frankenstein	3.1	A
Homo Genius	3.4	A
Hit Woman	3.1	A
Sorting Jack	3.3	A
Lacy Lowercase	3.4	A
Red Butler	3.1	A
Thomas Collins	2.1	U
Harry Krishna	2.9	U

Figure 5-1

The file, UNDRGRAD, is sorted into the file, VICTORS. The collating sequence is changed so the character, B, has a lower value than the character, A.

Calling SORTINITIAL and SORTINPUT

```
00001000 00000 0 $CONTROL USLIMIT
00002000 00000 0 << SPL EXAMPLE S2 >>
00003000 00000 0 << OUTPUTFILE SPECIFIED BUT NOT INPUTFILE >>
00004000 00000 0 << SORT THE FILE, MAIL2, INTO THE FILE, TEST >>
00005000 00000 0 << SORT ON ZIP CODES WITHIN STATES >>
00006000 00000 0 BEGIN
00007000 00000 1 BYTE ARRAY MAIL2(0:8):="MAIL2 ";
00008000 00004 1 BYTE ARRAY TEST(0:4):="TEST ";
00009000 00004 1 ARRAY ERROR(0:6):="ERROR IN SORT";
00010000 00007 1 ARRAY BUF(0:39);
00011000 00007 1 ARRAY KEYS(0:5);
00012000 00007 1 INTEGER OPIN, OPOUT, LEN;
00013000 00007 1 INTRINSIC FOPEN, FREAD, FPOINT, PRINT;
00014000 00007 1 INTRINSIC SORTINITIAL, SORTEND, SORTINPUT;
00015000 00007 1 << OPEN FILES >>
00016000 00007 1 OPIN:=FOPEN(MAIL2,%605,%305);
00017000 00010 1 OPOUT:=FOPEN(TEST,%605,%305);
00018000 00020 1 << ESTABLISH KEYS >>
00019000 00020 1 << MAJOR AT 52 FOR 2 BYTES (STATE) >>
00020000 00020 1 << MINOR AT 55 FOR 5 BYTES (ZIP CODE) >>
00021000 00020 1 KEYS(0):=52;
00022000 00023 1 KEYS(1):=2;
00023000 00026 1 KEYS(2):=0;
00024000 00031 1 KEYS(3):=55;
00025000 00034 1 KEYS(4):=5;
00026000 00037 1 KEYS(5):=0;
```

```

00027000 00042 1  << CALL SORTINITIAL - OUTPUT OPTION = 0 >>
00028000 00042 1  << INPUTFILE NOT SPECIFIED >>
00029000 00042 1      SORTINITIAL(,OPOUT,0,80,,2,KEYS);
00030000 00054 1      IF <> THEN GOTO ENDSORT;
00031000 00055 1  << READ RECORD FROM INPUT FILE >>
00032000 00055 1      INPUT:
00033000 00055 1          LEN:=FREAD(COPIN,BUF,-80);
00034000 00063 1          IF > THEN GOTO ENDSORT;
00035000 00064 1          BEGIN
00036000 00064 2  << CALL SORTINPUT >>
00037000 00064 2          SORTINPUT(BUF,LEN);
00038000 00067 2          IF <> THEN GOTO ENDSORT;
00039000 00070 2          GOTO INPUT
00040000 00076 2          END;
00041000 00076 1      ENDSORT:
00042000 00076 1          SORTEND;
00043000 00077 1          IF <> THEN GOTO SORTERR;
00044000 00100 1  << RESET OUTPUT FILE TO RECORD 1 >>
00045000 00100 1          FPOINT(OPOUT,0D);
00046000 00103 1      DISPLAY:
00047000 00103 1          LEN:=FREAD(OPOUT,BUF,40);
00048000 00111 1          IF > THEN GOTO STOP;
00049000 00112 1          PRINT(BUF,LEN,0);
00050000 00116 1          GOTO DISPLAY;
00051000 00117 1      SORTERR:
00052000 00117 1          PRINT(ERROR,7,0);
00053000 00123 1      STOP:
00054000 00123 1      END.

```



```

:BUILD TEST
:PREPRUN $OLDPASS; MAXDATA=4000

```

END OF PREPARE

SPACE	MANN	9999 GALAXY WAY	UNIVERSE	CA 61239	231-999-9999
KING	ARTHUR	329 EXCALIBUR ST	CAMELOT	CA 61322	812-200-0100
JENNA	GRANDTR	493 TWENTIETH ST	PROGRESSIVE	CA 61335	799-191-9191
SWASH	BUCKLER	497 PLAYACTING CT	MOVIETOWN	CA 61497	NONE
ALI	BABA	40 THIEVES WAY	SESAME	CO 69142	NONE
KARISSA	GRANDTR	7917 BROADMOOR WAY	BIGTOWN	MA 21799	713-244-3717
JANE	DOE	3959 TREEWOOD LN	BIGTOWN	MA 21843	714-399-4563
JOHN	DOUGHE	239 MAIN ST	HOMETOWN	MA 26999	714-411-1123
JAMES	DOE	4193 ANY ST	ANYTOWN	MD 00133	237-408-7100
KNEE	BUCKLER	974 FISTICUFF DR	PUGILIST	ND 04321	976-299-2990
JOHN	BIGTOWN	965 APPIAN WAY	METROPOLIS	NY 20013	619-407-2314
LOIS	ANYONE	6190 COURT ST	METROPOLIS	NY 20115	619-732-4997

Figure 5-2

The file, MAIL2, is sorted according to the major key, states, and the key, zip code, into the file, TEST.

Calling SORTOUTPUT

```

00001000 00000 0  $CONTROL USLINIT
00002000 00000 0  << SPL EXAMPLE S3 >>
00003000 00000 0  << INPUTFILE SPECIFIED BUT NOT OUTPUTFILE >>
00004000 00000 0  << SORT THE FILE, MAIL2 INTO THE FILE, TEST >>
00005000 00000 0  << SORT ON PHONE NUMBERS WITHIN STATES >>
00006000 00000 0  BEGIN
00007000 00000 1  BYTE ARRAY MAIL2(0:8):="MAIL2 ";
00008000 00004 1  BYTE ARRAY TEST(0:4):="TEST ";
00009000 00004 1  ARRAY ERROR(0:6):="ERROR IN SORT";
00010000 00007 1  ARRAY BUF(0:35);
00011000 00007 1  ARRAY KEYS(0:5);
00012000 00007 1  INTEGER OPIN,OPOUT,LEN:=36;
00013000 00007 1  INTRINSIC FOPEN,FREAD,FPOINT,PRINT,FWRITE;
00014000 00007 1  INTRINSIC SORTINITIAL,SORTEND,SORTOUTPUT;
00015000 00007 1  << OPEN FILES >>
00016000 00007 1  OPIN:=FOPEN(MAIL2,%605,%305);
00017000 00010 1  OPOUT:=FOPEN(TEST,%605,%305);
00018000 00020 1  << ESTABLISH KEYS >>
00019000 00020 1  << MAJOR AT 52 FOR 2 BYTES (STATE) >>
00020000 00020 1  << MINOR AT 61 FOR 12 BYTES (PHONE NO) >>
00021000 00020 1  KEYS(0):=52;
00022000 00023 1  KEYS(1):=2;
00023000 00026 1  KEYS(2):=0;
00024000 00031 1  KEYS(3):=61;
00025000 00034 1  KEYS(4):=12;
00026000 00037 1  KEYS(5):=0;
00027000 00042 1  << CALL SORTINITIAL - OUTPUT OPTION = 0 >>
00028000 00042 1  << OUTPUTFILE NOT SPECIFIED >>
00029000 00042 1  SORTINITIAL(OPIN,,0,,2,KEYS);
00030000 00053 1  IF <> THEN GOTO ENDSORT;
00031000 00054 1  << CALL SORTOUTPUT >>
00032000 00054 1  OUTPUT:
00033000 00054 1  SORTOUTPUT(BUF,LEN);
00034000 00057 1  IF <> THEN GOTO ENDSORT;
00035000 00060 1  IF LEN >=0 THEN
00036000 00063 1  BEGIN
00037000 00063 2  FWRITE(OPOUT,BUF,36,0);
00038000 00070 2  GOTO OUTPUT;
00039000 00076 2  END;
00040000 00076 1  << RESET OUTPUTFILE TO RECORD 1 >>
00041000 00076 1  FPOINT(OPOUT,0D);
00042000 00101 1  ENDSORT:
00043000 00101 1  SORTEND;
00044000 00102 1  IF <> THEN GOTO SORTERR;
00045000 00103 1  DISPLAY:
00046000 00103 1  LEN:=FREAD(OPOUT,BUF,36);
00047000 00111 1  IF > THEN GOTO STOP;
00048000 00112 1  PRINT(BUF,LEN,0);
00049000 00116 1  GOTO DISPLAY;
00050000 00117 1  SORTERR:
00051000 00117 1  PRINT(ERROR,7,0);
00052000 00123 1  STOP:
00053000 00123 1  END.

```

```
:BUILD TEST
:PREPRUN $OLDPASS; MAXDATA=15000
```

END OF PREPARE

SPACE	MANN	9999 GALAXY WAY	UNIVERSE	CA 61239	231-999-9999
JENNA	GRANDTR	493 TWENTIETH ST	PROGRESSIVE	CA 61335	799-191-9191
KING	ARTHUR	329 EXCALIBUR ST	CAMELOT	CA 61322	812-200-0100
SWASH	BUCKLER	497 PLAYACTING CT	MOVIETOWN	CA 61497	NONE
ALI	BABA	40 THIEVES WAY	SESAME	CO 69142	NONE
KARISSA	GRANDTR	7917 BROADMOOR WAY	BIGTOWN	MA 21799	713-244-3717
JANE	DOE	3959 TREEWOOD LN	BIGTOWN	MA 21843	714-399-4563
JOHN	DOUGHE	239 MAIN ST	HOMETOWN	MA 26999	714-411-1123
JAMES	DOE	4193 ANY ST	ANYTOWN	MD 00133	237-408-7100
KNEE	BUCKLER	974 FISTICUFF DR	PUGILIST	ND 04321	976-299-2990
JOHN	BIGTOWN	965 APPIAN WAY	METROPOLIS	NY 20013	619-407-2314
LOIS	ANYONE	6190 COURT ST	METROPOLIS	NY 20115	619-732-4997

Figure 5-3

The file, MAIL2, is sorted according to the major key, states, and the key, phone numbers, into the file, TEST.

Calling SORTINITIAL without the *inputfile* and *outputfile* parameters

```
00001000 00000 0 $CONTROL USLINIT
00002000 00000 0 << SPL EXAMPLE S4 >>
00003000 00000 0 << NEITHER INPUTFILE NOR OUTPUTFILE SPECIFIED >>
00004000 00000 0 << SORT MAIL2 INTO TEST >>
00005000 00000 0 << SORT ON FIRST NAMES WITHIN LAST NAMES >>
00006000 00000 0 BEGIN
00007000 00000 1 BYTE ARRAY MAIL2 (0:8):="MAIL2 ";
00008000 00004 1 BYTE ARRAY TEST(0:4):="TEST ";
00009000 00004 1 ARRAY ERROR(0:6):="ERROR IN SORT";
00010000 00007 1 ARRAY BUF(0:39);
00011000 00007 1 ARRAY KEYS(0:5);
00012000 00007 1 INTEGER OPIN, OPOUT, LEN;
00013000 00007 1 INTRINSIC FOPEN, FREAD, FWRITE, FPOINT, PRINT;
00014000 00007 1 INTRINSIC SORTINITIAL, SORTEND;
00015000 00007 1 INTRINSIC SORTINPUT, SORTOUTPUT;
00016000 00007 1 << OPEN FILES >>
00017000 00007 1 OPIN:=FOPEN(MAIL2,%605,%305);
00018000 00010 1 OPOUT:=FOPEN(TEST,%605,%305);
00019000 00020 1 << ESTABLISH KEYS >>
00020000 00020 1 << MAJOR AT 11 FOR 9 BYTES (LAST NAME) >>
00021000 00020 1 << MINOR AT 1 FOR 10 BYTES (FIRST NAME) >>
00022000 00020 1 KEYS(0):=11;
00023000 00023 1 KEYS(1):=9;
00024000 00026 1 KEYS(2):=0;
00025000 00031 1 KEYS(3):=1;
00026000 00034 1 KEYS(4):=10;
00027000 00037 1 KEYS(5):=0;
```



```

00028000 00042 1  << CALL SORTINITIAL - OUTPUT OPTION = 0 >>
00029000 00042 1  << INPUTFILE AND OUTPUTFILE NOT SPECIFIED >>
00030000 00042 1      SORTINITIAL(,,0,80,,2,KEYS);
00031000 00053 1      IF <> THEN GOTO ENDSORT;
00032000 00054 1  << READ RECORD FROM INPUT FILE >>
00033000 00054 1      INPUT:
00034000 00054 1          LEN:=FREAD(OPIN,BUF,-80);
00035000 00062 1          IF > THEN GOTO OUTPUT;
00036000 00063 1  << CALL SORTINPUT >>
00037000 00063 1      BEGIN
00038000 00063 2          SORTINPUT(BUF,LEN);
00039000 00066 2          IF <> THEN GOTO ENDSORT;
00040000 00067 2      END;
00041000 00067 1          GOTO INPUT;
00042000 00075 1  << CALL SORTOUTPUT >>
00043000 00075 1      OUTPUT:
00044000 00075 1      BEGIN
00045000 00075 2          SORTOUTPUT(BUF,LEN);
00046000 00100 2          IF <> THEN GOTO ENDSORT;
00047000 00101 2          IF LEN >=0 THEN
00048000 00104 2              BEGIN
00049000 00104 3                  FWRITE(OPOUT,BUF,40,0);
00050000 00111 3                  GOTO OUTPUT;
00051000 00112 3              END;
00052000 00112 2      END;
00053000 00112 1  ENDSORT:
00054000 00112 1      SORTEND;
00055000 00113 1          IF <> THEN GOTO SORTERR;
00056000 00114 1  << RESET OUTPUTFILE TO FIRST RECORD >>
00057000 00114 1          FPOINT(OPOUT,0D);
00058000 00117 1      DISPLAY:
00059000 00117 1          LEN:=FREAD(OPOUT,BUF,40);
00060000 00125 1          IF > THEN GOTO STOP;
00061000 00126 1          PRINT(BUF,LEN,0);
00062000 00132 1          GOTO DISPLAY;
00063000 00133 1  SORTERR:
00064000 00133 1          PRINT(ERROR,7,0);
00065000 00137 1  STOP:
00066000 00137 1  END.

```

```
:BUILD TEST
:PREPRUN $OLDPASS; MAXDATA=15000
```

```
END OF PREPARE
```

```
LOIS      ANYONE   6190 COURT ST      METROPOLIS NY 20115 619-732-4997
KING      ARTHUR   329 EXCALIBUR ST   CAMELOT    CA 61322 812-200-0100
ALI       BABA      40 THIEVES WAY     SESAME     CO 69142 NONE
JOHN      BIGTOWN   965 APPIAN WAY     METROPOLIS NY 20013 619-407-2314
KNEE     BUCKLER   974 FISTICUFF DR   PUGILIST   ND 04321 976-299-2990
SWASH    BUCKLER   497 PLAYACTING CT  MOVIE TOWN CA 61497 NONE
JAMES    DOE       4193 ANY ST        ANYTOWN    MD 00133 237-408-7100
JANE     DOE       3959 TREEWOOD LN   BIGTOWN    MA 21843 714-399-4563
JOHN     DOUGHE    239 MAIN ST        HOMETOWN   MA 26999 714-411-1123
JENNA    GRANDTR   493 TWENTIETH ST   PROGRESSIVE CA 61335 799-191-9191
KARISSA  GRANDTR   7917 BROADMOOR WAY BIGTOWN    MA 21799 713-244-3717
SPACE    MANN      9999 GALAXY WAY    UNIVERSE   CA 61239 231-999-9999
```

Figure 5-4

The file, MAIL2, is sorted according to the major key, last name, and the key, first name, into the file, TEST.

Calling SORTINIT with the *charseq* parameter.

```
00000 0  $CONTROL USLINIT
00000 0  << SPL EXAMPLE S5 >>
00000 0  << SPL CHARSEQ PARAMETER EXAMPLE >>
00000 0  << INPUT AND OUTPUT FILE SPECIFIED >>
00000 0  << SORT THE FILE, MISCDATA, INTO THE FILE, SORTDATA >>
00000 0  << SORT ON THE FIRST 5 CHARACTERS USING THE COLLATING SEQUENCE
00000 0  FOR THE NATIVE LANGUAGE, AMERICAN >>
00000 0
00000 0  BEGIN
00000 1  BYTE ARRAY INPUT(0:9) := "MISCDATA ";
00006 1  BYTE ARRAY OUTPUT(0:9) := "SORTDATA ";
00006 1  ARRAY BUF(0:21);
00006 1  INTEGER ARRAY IN(0:1), OUT(0:1);
00006 1  INTEGER LEN;
00006 1  INTEGER NUMKEYS := 1;
00006 1  INTEGER ARRAY KEYS(0:2);
00006 1  INTEGER ARRAY CSEQ(0:1);
00006 1  INTRINSIC FPOINT, QUIT, FOPEN, FREAD, PRINT, SORTINIT, SORTEND;
00006 1
00006 1  IN(0) := FOPEN(INPUT,5);
00010 1  IN(1) := 0;
00013 1  IF <> THEN QUIT(90);
00016 1  OUT(0) := FOPEN(OUTPUT,4,4);
00027 1  OUT(1) := 0;
00032 1  IF <> THEN QUIT(91);
00035 1  << ESTABLISH THE KEY WITH KEY TYPE CHARACTER >>
00035 1  KEYS(0) := 1;
00040 1  KEYS(1) := 5;
00043 1  KEYS(2) := 9;
00046 1  << ESTABLISH THE CHARSEQ PARAMETER; LANGUAGE = AMERICAN
00046 1  LANGUAGE ID = 1 >>
00046 1  CSEQ(0) := 1;
00051 1  CSEQ(1) := 1;
00054 1  << CALL SORTINIT WITH THE CHARSEQ PARAMETER >>
00054 1  SORTINIT(IN,OUT,,,NUMKEYS,KEYS,,,,,CSEQ);
00066 1  SORTEND;
00067 1  FPOINT(OUT,00);
```

```

00072 1 LOOP:   LEN := FREAD(OUT, BUF, 21);
00100 1 IF <> THEN QUIT(93);
00103 1 PRINT(BUF, LEN, 0);
00107 1 GOTO LOOP;
00113 1 END.

```

```

END OF COMPILE
:BUILD SORTDATA;REC=-80,,F,ASCII
:PREPRUN $DLOPASS;MAXDATA=30000

```

END OF PREPARE

```

Cleaver Wallace
David Henry
Henry John
James Dennis
James Henry
Wallace George

```

Figure 5-5

Multirecord, NOBUF, and Buffered Files

```
:SPL NOBUF3
```

```
PAGE 0001 HEWLETT-PACKARD 32100A.08.0C SPL[4W] TUE, JAN 29, 1980, 11:09 AM C
```

```

00000 0 $CONTROL USLINIT
00000 0 This program demonstrates the use of multirecord, NOBUF, and
00000 0 buffered files and the errorparm with SORT.
00000 0
00000 0 BEGIN
00000 1     INTRINSIC FOPEN,FCLOSE,
00000 1             SORTINIT,SORTEND,SORTSTAT,SORTERRORMESS,
00000 1             ASCII,PRINT;
00000 1     INTEGER ARRAY INPUT(0:3),
00000 1             OUTPUT(0:1),
00000 1             KEYS(0:2),
00000 1             STATISTICS(0:11);
00000 1     INTEGER ERROR,
00000 1             LENGTH,
00000 1             RECSIZE := 80;
00000 1     LOGICAL FAILURE;
00000 1     BYTE ARRAY INFILE1(0:8) := ``UNSORT01 ``;
00006 1     BYTE ARRAY INFILE2(0:8) := ``UNSORT02 ``;
00006 1     BYTE ARRAY INFILE3(0:8) := ``UNSORT03 ``;
00006 1     BYTE ARRAY OUTFILE(0:8) := ``SORTED ``;
00006 1     ARRAY WORD'BUF(0:40);
00006 1     BYTE ARRAY BUFFER(*) = WORD'BUF;
00006 1
00006 1     EQUATE NEWFILE      = %0,
00006 1             OLDFILE    = %3,
00006 1             READ       = %0,
00006 1             WRITE      = %1,
00006 1             MULTIRECORD = %20,
00006 1             NOBUF      = %400

```

```

00006 1          BUFFERED      = %0,
00006 1          NO'CHANGE     = %0,
00006 1          SAVE'PERM      = %1,
00006 1          RETURN'SPACE= %10,
00006 1          UNRESTRICTED= %0;
00006 1
00006 1  <<ESTABLISH THE FILE ARRAYS FOR INPUT AND OUTPUT>>
00006 1  INPUT(0) := FOPEN(INFILE1,OLDFILE,MULTIRECORD+READ);
00012 1  INPUT(1) := FOPEN(INFILE2,OLDFILE,NOBUF+READ);
00024 1  INPUT(2) := FOPEN(INFILE3,OLDFILE,BUFFERED+READ);
00036 1  INPUT(3) := 0;
00041 1
00041 1  OUTPUT(0) := FOPEN(OUTFILE,NEWFILE,MULTIRECORD+WRITE,-RECSIZE)
00055 1  OUTPUT(1) := 0;
00060 1
00060 1  <<ESTABLISH THE KEYS ARRAY>>
00060 1  KEYS(0) := 73;  << POSITION >>
00063 1  KEYS(1) := 8;   << LENGTH  >>
00066 1  KEYS(2) := 0;   << ASCENDING, TYPE BYTE OR LOGICAL >>
00071 1
00071 1  <<OTHER INITIALIZATIONS>>
00071 1  FAILURE := FALSE;
00073 1  ERROR := TRUE;
00075 1
00075 1  <<PERFORM THE SORT AND CHECK FOR ERRORS>>
00075 1
00075 1  SORTINIT(INPUT,OUTPUT,0,RECSIZE,,1,KEYS,,,STATISTICS,FAILURE)
00113 1  SORTEND;
00114 1
00114 1  IF FAILURE
00114 1  THEN BEGIN << PRINT THE ERROR MESSAGE AND NUMBER >>
00116 2    SORTERRORMESS(ERROR,BUFFER,LENGTH);
00122 2    PRINT(WORD'BUF,-LENGTH,%320);
00127 2    MOVE BUFFER := `` ( '';
00144 2    LENGTH := ASCII(ERROR,10,BUFFER(3)) + 3;
00154 2    MOVE BUFFER(LENGTH) := `` )'';
00164 2    LENGTH := LENGTH + 2;
00167 2    PRINT(WORD'BUF,-LENGTH, %40);
00174 2    END
00174 1  ELSE      << PRINT THE STATISTICS >>
00175 1    SORTSTAT(STATISTICS);
00177 1
00177 1  FCLOSE(INPUT(0),NO'CHANGE,UNRESTRICTED);
00203 1  FCLOSE(INPUT(1),NO'CHANGE,UNRESTRICTED);
00207 1  FCLOSE(INPUT(2),NO'CHANGE,UNRESTRICTED);
00213 1
00213 1  FCLOSE(OUTPUT(0),SAVE'PERM+RETURN'SPACE,UNRESTRICTED);
00221 1
00221 1  END.
PRIMARY DB STORAGE=%016;    SECONDARY DB STORAGE=%00122
NO. ERRORS=0000;           NO. WARNINGS=0000
PROCESSOR TIME=0:00:02;    ELAPSED TIME=0:00:08

```

END OF COMPILE
:PREP \$OLDPASS,PNOBUF3;MAXDATA=31232

END OF PREPARE
:RUN PNOBUF3;LIB=G

STATISTICS

NUMBER OF RECORDS =	150
NUMBER OF INTERMEDIATE PASSES =	0
SPACE AVAILABLE (IN WORDS) =	14,085
NUMBER OF COMPARES =	1,170
NUMBER OF SCRATCHFILE IO'S =	102
CPU TIME (MINUTES) =	.03
ELAPSED TIME (MINUTES) =	.10

END OF PROGRAM

CALLING MERGE FROM A SPL/3000 PROGRAM

SECTION

VI

You can merge two or more sorted files from an SPL/3000 program by using intrinsic calls. These intrinsics (SPL/3000 procedures) are part of SORT-MERGE/3000 and are called by using the SYSTEM INTRINSIC declarations in your program. The various parameters of these intrinsics are used by SORT-MERGE/3000 to perform specific operations.

The MERGE program intrinsics

The following is a list of the MERGE program intrinsics which reside in the MERGELIB segment of the system segmented library:

INTRINSIC	DESCRIPTION
MERGEINIT	Merges two or more sorted files.
MERGEOUTPUT	Requests records from MERGEINIT, one at a time, if the <i>outputfiles</i> parameter is not specified in MERGEINIT.
MERGEEND	Restores the data stack to its original state. MERGEEND must be called only if MERGEINIT is called.
MERGESTAT	Prints the MERGE statistics on \$STDLIST.
MERGETITLE	Prints the version number and title of the MERGELIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.
MERGEERRORMESS	Called to retrieve and print a message if a fatal error occurs during MERGE. MERGEERRORMESS is called from a user supplied error procedure (the <i>errorproc</i> parameter of MERGEINIT).

The MERGEINIT intrinsic initiates the MERGE operation. After calling MERGEINIT, you should call MERGEOUTPUT if the *outputfiles* parameter of MERGEINIT is not specified. This is followed by a call to the MERGEEND intrinsic. MERGEINIT and MERGEEND must be called by the same procedure. Call MERGESTAT if you want the display of the MERGE statistics. Additionally, call SORTERRORMESS from the user supplied procedure, *errorproc*, if you want a display of the message when the error occurs. MERGEOUTPUT, MERGESTAT, and MERGEERRORMESS are optional but their order is important if they are called. The optional intrinsic, MERGETITLE, can be called at any stage. The following flowchart illustrates the MERGE operation when MERGEOUTPUT and MERGESTAT are called:

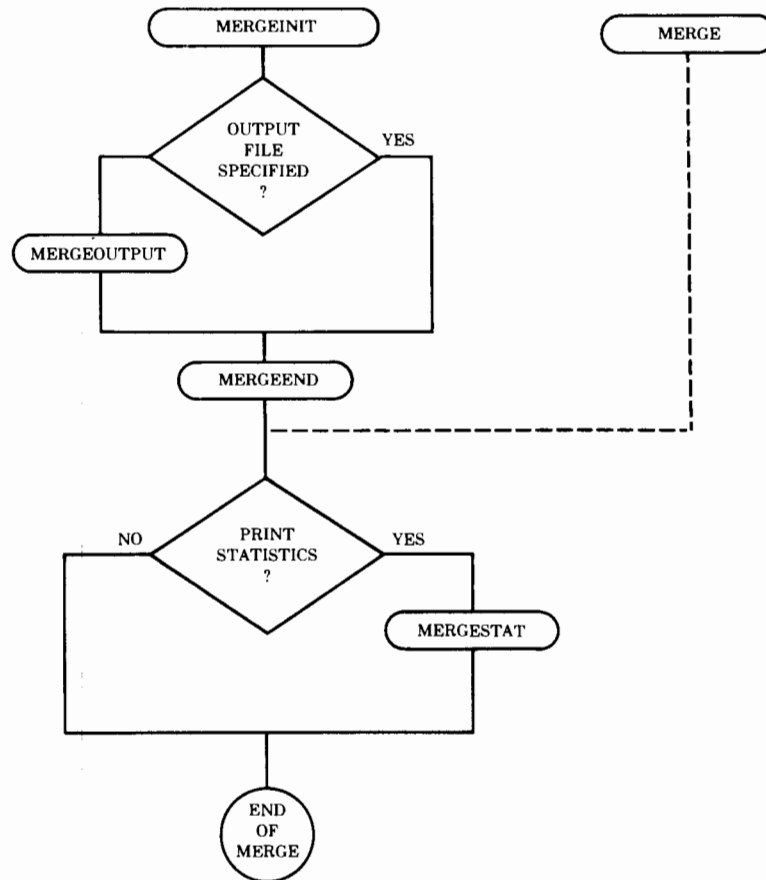


Figure 6-1. Flowchart of Merge Intrinsics

Preparation and Execution of the MERGE programs

The amount of available memory can affect the time required to perform a MERGE. MERGE programs with files opened multirecord should be prepared with the maximum available segsize which is specified by the MAXDATA=segsize parameter of the :PREP or :RUN commands. MERGE programs with files opened NOBUF should increase the segsize, allowing one block per file.

If the error message INSUFFICIENT STACK SPACE is displayed, increase the MAXDATA parameter. If the message TOO MANY FILES OPEN (FSERR 71) appears, it means MPE has no room for its tables in the user data segment. Use the NOCB parameter of the :RUN command during the execution of the program in this case.

NOTE: The MERGE intrinsic is included in this manual for the maintenance of existing SPL/3000 programs.

MERGEINIT

Merges two or more sorted files.

SYNTAX

```
PROCEDURE MERGEINIT (IA inputfiles, P preprocessor, IA outputfiles,  
P postprocessor, LV keyonly, IV numkeys, IA keys,  
IA altseq, LP keycompare, P errorproc, IA statistics,  
L failure, I errorparm, I spaceallocation, IA charseq, O-V parm1);
```

PARAMETERS

- inputfiles* An integer array containing the file identifications of the input files to be merged. The array is terminated with a word of zero. If the files are opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking. \$Null is not a valid input file.
- preprocessor* A procedure called whenever a record is read from the input file. The call should include a statement of the following form:
- ```
PROCEDURE preprocessor (file, record, length);
```
- file* is an integer passed by reference which is the index to the *inputfiles* array of the file from which the record is read. The value of the *file* parameter lies between 0 and the number of input files minus one. *record* is a byte array denoting the data record. *length* is an integer passed by reference denoting the number of characters in the record.
- outputfiles* An integer array containing the file identification of the output file. The second word must contain a zero to indicate the end of the list. If the file is opened with either the NOBUF or MR (multirecord) access option (aoption), SORT or MERGE will perform the buffering and blocking/deblocking.



*postprocessor*

A procedure called before each record is sent to the output file. Either this parameter or *outputfiles* (or both) must be specified. *postprocessor* is called by a statement of the following form:

```
PROCEDURE postprocessor (record, length);
```

*record* is a byte array which is the data record. *length* is an integer passed by reference denoting the number of characters in the record.

*keyonly*

A logical procedure, which if true, passed by value causes only the key fields; concatenated together with the major key on the left followed by other keys; sent as output. The *keycompare* parameter must not be specified in this case. If *keyonly* is false, the entire records are sent as output. The default for *keyonly* is false.

*numkeys* and *keys*

*numkeys* is an integer passed by value and *keys* is an integer array. They describe the way records are merged. If either is specified, the other must also be specified and *keycompare* must not be specified. *numkeys* denotes the number of keys used during the comparison of records. It may be either equal to or greater than one. *keys* specifies the way the records are compared. For each key being specified, *keys* contains three words:

First word gives the position of the first character of the key within the record. Second word denotes the number of characters in the key. Third word (bits 0 through 7) gives the ordering sequence of the records. (0 for ascending, 1 for descending)

bits 8 through 15 of the third word indicate the type of data according to the following convention:

0=logical or byte (same as the *type*, BYTE, in interactive mode)

1=two's complement (including integer and double integer)

2=floating point

3=packed decimal

5=packed decimal with even number of digits

4=Display-Trailing-Sign (see the KEY command in Section II)

6=Display-Leading-Sign

7=Display-Leading-Sign-Separate

8=Display-Trailing-Sign-Separate

9=Character (Collating sequence of *charseq*[see below] is used)

**Note:** The integrity of the integer array *keys* must be maintained throughout the merge operation. Do not change the array until after a call has been made to the intrinsic MERGEEND.

*altseq* An integer array defining an alternate collating sequence. The first character (bits 0-7) of the array is defined according to Table 3-1. The second character (bits 8-15) specifies one less than the total number of characters in the collating sequence (in this case 255 or %377). These two characters are followed by the actual collating sequence responsible for the particular MERGE operation.

*keycompare* A logical procedure specified if you do not specify *numkeys* and *keys*. It is called whenever two records are compared. This call should include a statement of the following form:

```
LOGICAL PROCEDURE keycompare (rec1, len1, rec2, len2);
```

*rec1* and *rec2* are byte arrays and are pointers to the two records. *len1* and *len2* are integers passed by value and are the lengths of the records in characters. *keycompare* returns a true value if *rec1* precedes *rec2*, and a false value otherwise. It returns a true value even in the case of ties. This ensures that the original sequence is preserved in the case of ties. those specified later.

*errorproc* A procedure used in conjunction with the MERGEERRORMESS procedure. It is called as follows whenever a fatal error occurs during MERGE:

```
PROCEDURE errorproc (errorcode);
```

*errorcode* is an integer passed by reference and is the MERGE program error number. It is passed to *errorproc* when an error occurs. If *errorproc* or *errorparm* are not specified, a default procedure is used which prints the error message corresponding to the particular *errorcode*. For a list of these error messages, see Appendix A.

*statistics* An integer array which if specified, gives the following data:

Zeroth word=  
number of input files.

First and second words=  
number of merged records (double integer)

Third word=  
space available for merging.

Fourth and fifth words=  
number of comparisons (double integer).

Sixth and seventh words=  
CPU time (in milliseconds, double integer).

Eighth and ninth words=  
elapsed time (in milliseconds, double integer).

*failure* A logical word passed by reference which is set to -1 (true) if a fatal error occurs, and 0 (false) otherwise.

Error conditions:

CCE=  
no error occurred (*failure* set to false)

CCG=  
an error occurred (*failure* set to true)

*errorparm* An integer variable which, if specified, is set to the MERGELIB error number if an error occurs. The MERGEERRORMESS intrinsic can be used to obtain the error message text. If the errorparm is supplied, the errorproc procedure is ignored and no error messages are display. For a list of error messages see Appendix A.

*spaceallocation* An integer variable which, if specified, is used to determine stack allocation. A positive spaceallocation specifies the number of words that may be used for sorting and buffering. A negative values specifies the number of words that should be left for the user after determining the amount available. Zero will cause a default value to be used.

*charseq* 2 word integer array. Set word 0 to 1. Set word 1 to *langid* of the native language whose collating sequence is to be used to sort keys of type 9 (character).

*parm1* unused

# MERGEOUTPUT

Requests records from MERGEINIT, one at a time, if the *outputfiles* parameter is not specified in MERGEINIT.

## SYNTAX

```
PROCEDURE MERGEOUTPUT (LA record, I length);
```

## PARAMETERS

*record*                    A logical array receiving the next output record.

*length*                    An integer passed by reference denoting the number of characters in the record.

MERGEOUTPUT is called after MERGEINIT but before MERGEEND.

# MERGEEND

Restores the data stack to its original state.

## SYNTAX

**PROCEDURE MERGEEND;**

It must be called only if MERGEINIT is called, and it must be called from the same subroutine that called MERGEINIT.

# MERGESTAT

Prints the MERGE statistics on \$STDLIST.

## SYNTAX

*IA*  
**MERGESTAT ( *statistics* );**

*statistics* is an integer array. MERGESTAT is called after MERGEEND.



# **MERGETITLE**

Prints the version number and title of the MERGELIB segment along with the date and time produced by the DATELINE intrinsic on \$STDLIST.

## **SYNTAX**

**PROCEDURE MERGETITLE;**

This intrinsic can be called from the program at any stage after the system intrinsics are declared.

# MERGEERRORMESS

Called to retrieve and print a message if a fatal error occurs during MERGE. MERGEERRORMESS is called from a user supplied error procedure (the *errorproc* parameter of MERGEINIT).

## SYNTAX

```
PROCEDURE MERGEERRORMESS (IV BA I
 (errorcode, message, length);
```

## PARAMETERS

|                  |                                                                                                                            |
|------------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>errorcode</i> | An integer passed by value denoting the MERGE program error number and is passed to <i>errorproc</i> when an error occurs. |
| <i>message</i>   | A byte array containing the text of the message. The <i>message</i> parameter must be at least 72 characters long.         |
| <i>length</i>    | An integer passed by reference denoting the length of the message in characters.                                           |

MERGEERRORMESS converts *errorcode* values into ASCII strings. It works in conjunction with the *errorproc* parameter of MERGEINIT.



# MERGE

Initiates the MERGE operation (to be used only for existing SPL/3000 programs).

## SYNTAX

```
PROCEDURE MERGE (IV numinputfiles, IA inputfiles, IV outputfile,
IV keyonly, IV numkeys, IA keys, P preprocessor,
P postprocessor, P errorproc, LP keycompare, IA statistics,
L failure); O-V
```

## PARAMETERS

- numinputfiles* An integer passed by value denoting the number of input files to be merged. This parameter is not optional and is either equal to or greater than one.
- inputfiles* An integer array containing the MPE/3000 file numbers of the files to be merged. These file numbers appear in the locations *inputfiles*(0) through *inputfiles*(*numinputfiles*-1). This parameter is not optional.
- outputfile* Unlike MERGEINIT, where the *outputfiles* parameter is an integer array, *outputfile* is an integer passed by value specifying the MPE/3000 file number of the file on which the merged records are written. If *outputfile* is not specified, the records are not written anywhere. In this case, *postprocessor* must be specified.

All the other parameters are similar to the MERGEINIT parameters except the positions of the parameters, *errorproc* and *keycompare*, are interchanged. MERGE is less powerful than MERGEINIT in that it does not have the *altseq* parameter. Also, MERGEOUTPUT and MERGEEND must not be called when MERGE is called.

## Calling MERGE (Example)

```
00001000 00000 0 $CONTROL USLINIT
00002000 00000 0 << SPL EXAMPLE S5 >>
00003000 00000 0 << MERGE THE SORTED FILES, MAIL1 AND MAIL2, >>
00004000 00000 0 << INTO THE >>
00005000 00000 0 << FILE, TEST1. >>
00006000 00000 0 BEGIN
00007000 00000 1 BYTE ARRAY MAIL1(0:5):="MAIL1 ";
00008000 00004 1 BYTE ARRAY MAIL2(0:4):="MAIL2 ";
00009000 00004 1 BYTE ARRAY TEST1(0:4):="TEST1 ";
00010000 00004 1 ARRAY ERROR(0:6):="ERROR IN MERGE";
00011000 00007 1 ARRAY BUF(0:35);
00012000 00007 1 ARRAY KEYS(0:5);
00013000 00007 1 INTEGER ARRAY INFILES(0:1);
00014000 00007 1 INTEGER OPOUT,LEN;
00015000 00007 1 LOGICAL FAILURE;
00016000 00007 1 INTRINSIC FOPEN,FREAD,FPOINT,PRINT,MERGE;
```

```

00017000 00007 1 << OPEN FILES >>
00018000 00007 1 INFILES(0):=FOPEN(MAIL1,%605,%305);
00019000 00011 1 INFILES(1):=FOPEN(MAIL2,%605,%305);
00020000 00022 1 OPOUT:=FOPEN(TEST1,%605,%305);
00021000 00032 1 << ESTABLISH THE KEYS >>
00022000 00032 1 << MAJOR AT 11 FOR 9 BYTES (LAST NAME) >>
00023000 00032 1 << MINOR AT 1 FOR 10 BYTES (FIRST NAME) >>
00024000 00032 1 KEYS(0):=11;
00025000 00035 1 KEYS(1):=9;
00026000 00040 1 KEYS(2):=0;
00027000 00043 1 KEYS(3):=1;
00028000 00046 1 KEYS(4):=10;
00029000 00051 1 KEYS(5):=0;
00030000 00054 1 << CALL MERGE >>
00031000 00054 1 MERGE(2,INFILES,OPOUT,,2,KEYS);
00032000 00065 1 IF <> THEN GOTO MERGERR;
00033000 00066 1 << OUTPUT MERGED FILE >>
00034000 00066 1 << RESET OUTPUTFILE TO RECORD 1 >>
00035000 00066 1 FPOINT(OPOUT,0D);
00036000 00071 1 DISPLAY:
00037000 00071 1 BEGIN
00038000 00071 2 LEN:=FREAD(OPOUT,BUF,36);
00039000 00077 2 IF > THEN GOTO STOP;
00040000 00100 2 PRINT(BUF,LEN,0);
00041000 00104 2 GOTO DISPLAY;
00042000 00114 2 END;
00043000 00114 1 MERGERR:
00044000 00114 1 PRINT(ERROR,7,0);
00045000 00120 1 STOP:
00046000 00120 1 END.

```

:BUILD TEST1

:PREPRUN \$OLDPASS; MAXDATA=15000

END OF PREPARE

|           |          |                  |           |             |    |       |              |
|-----------|----------|------------------|-----------|-------------|----|-------|--------------|
| PLAINS    | ANTELOPE | 201 OPENS        | SPACE AVE | BIGCOUNTRY  | WY | 49301 | 369-732-4821 |
| LOIS      | ANYONE   | 6190 COURT       | ST        | METROPOLIS  | NY | 20115 | 619-732-4997 |
| KING      | ARTHUR   | 329 EXCALIBUR    | ST        | CAMELOT     | CA | 61322 | 812-200-0100 |
| ALI       | BABA     | 40 THIEVES       | WAY       | SESAME      | CO | 69142 | NONE         |
| BLACK     | BEAR     | 47 ALLOVER       | DR        | ANYWHERE    | US | 00111 | NONE         |
| JOHN      | BIGTOWN  | 965 APPIAN       | WAY       | METROPOLIS  | NY | 20013 | 619-407-2314 |
| KNEE      | BUCKLER  | 974 FISTICUFF    | DR        | PUGILIST    | ND | 04321 | 976-299-2990 |
| SWASH     | BUCKLER  | 497 PLAYACTING   | CT        | MOVIETOWN   | CA | 61497 | NONE         |
| ANIMAL    | CRACKERS | 1000 ANYWHERE    | PL        | ALLOVER     | US | 00001 | 001-100-1000 |
| MULE      | DEER     | 963 FOREST       | PL        | NICECOUNTRY | CA | 97643 | 493-900-9000 |
| WHITETAIL | DEER     | 34 WOODSY        | PL        | BACKCOUNTRY | ME | 01341 | 619-433-4333 |
| JAMES     | DOE      | 4193 ANY         | ST        | ANYTOWN     | MD | 00133 | 237-408-7100 |
| JANE      | DOE      | 3959 TREEWOOD    | LN        | BIGTOWN     | MA | 21843 | 714-399-4563 |
| PRAIRE    | DOG      | 493 ROLLINGHILLS | DR        | OPENS       | ND | 24321 | 992-419-4192 |
| JOHN      | DOUGHE   | 239 MAIN         | ST        | HOMETOWN    | MA | 26999 | 714-411-1123 |
| MALLARD   | DUCK     | 79 MARSH         | PL        | PUDDLEDUCK  | CA | 97432 | 492-492-4922 |
| JENNA     | GRANDTR  | 493 TWENTIETH    | ST        | PROGRESSIVE | CA | 61335 | 799-191-9191 |
| KARISSA   | GRANDTR  | 7917 BROADMOOR   | WAY       | BIGTOWN     | MA | 21799 | 713-244-3717 |
| SNOWSHOE  | HARE     | 742 FRIGID       | WAY       | COLDSPOT    | MN | 37434 | 732-732-7320 |
| MOUNTAIN  | LION     | 796 KING         | DR        | THICKET     | NM | 37643 | 712-712-7122 |
| SPACE     | MANN     | 9999 GALAXY      | WAY       | UNIVERSE    | CA | 61239 | 231-999-9999 |
| SWAMP     | RABBIT   | 4444 DAMPPLACE   | RD        | BAYOU       | LA | 79999 | NONE         |
| NASTY     | RATTLER  | 243 DANGER       | AVE       | DESERTVILLE | CA | 87654 | 828-432-4321 |
| BIGHORN   | SHEEP    | 999 MOUNTAIN     | DR        | HIGHPLACE   | CO | 34567 | 776-409-9040 |
| GREY      | SQUIRREL | 432 PLEASANT     | DR        | FALLCOLORS  | MA | 14321 | 619-619-6199 |

Figure 6-2

The files, MAIL1 and MAIL2, are merged into the file, TEST.

# EXAMPLES

Merging files opened MR and NOBUF.

:SPL NOBUF4

PAGE 0001 HEWLETT-PACKARD 32100A.08.0C SPL[4W] TUE, JAN 29, 1980, 11:10 AM (

```
00000 0 $CONTROL USLINIT
00000 0 This program demonstrates the use of multirecord, NOBUF, and
00000 0 buffered files and the errorparm with MERGEINIT.
00000 0
00000 0 BEGIN
00000 1 INTRINSIC FOPEN,FCLOSE,
00000 1 MERGEINIT,MERGEEND,MERGESTAT,MERGEERRORMESS,
00000 1 ASCII,PRINT;
00000 1 INTEGER ARRAY INPUT(0:3),
00000 1 OUTPUT(0:1),
00000 1 KEYS(0:2),
00000 1 STATISTICS(0:11);
00000 1 INTEGER ERROR,
00000 1 LENGTH,
00000 1 RECSIZE := 80;
00000 1 LOGICAL FAILURE;
00000 1 BYTE ARRAY INFILE1(0:8) := ``SORTED01 ``;
00006 1 BYTE ARRAY INFILE2(0:8) := ``SORTED02 ``;
00006 1 BYTE ARRAY INFILE3(0:8) := ``SORTED03 ``;
00006 1 BYTE ARRAY OUTFILE(0:8) := ``MERGED ``;
00006 1 ARRAY WORD'BUF(0:40);
00006 1 BYTE ARRAY BUFFER(*) = WORD'BUF;
00006 1
00006 1 EQUATE NEWFILE = %0,
00006 1 OLDFILE = %3,
00006 1 READ = %0,
00006 1 WRITE = %1,
00006 1 MULTIRECORD = %20,
00006 1 NOBUF = %400,
00006 1 BUFFERED = %0,
00006 1 NO'CHANGE = %0,
00006 1 SAVE'PERM = %1,
00006 1 RETURN'SPACE= %10,
00006 1 UNRESTRICTED= %0;
00006 1
00006 1 <<ESTABLISH THE FILE ARRAYS FOR INPUT AND OUTPUT>>
00006 1 INPUT(0) := FOPEN(INFILE1,OLDFILE,MULTIRECORD+READ);
00012 1 INPUT(1) := FOPEN(INFILE2,OLDFILE,NOBUF+READ);
00024 1 INPUT(2) := FOPEN(INFILE3,OLDFILE,BUFFERED+READ);
00036 1 INPUT(3) := 0;
00041 1
00041 1 OUTPUT(0) := FOPEN(OUTFILE,NEWFILE,MULTIRECORD+WRITE,-RECSIZE)
00055 1 OUTPUT(1) := 0;
00060 1
00060 1 <<ESTABLISH THE KEYS ARRAY>>
00060 1 KEYS(0) := 73; << POSITION >>
00063 1 KEYS(1) := 8; << LENGTH >>
00066 1 KEYS(2) := 0; << ASCENDING, TYPE BYTE OR LOGICAL >>
```

```

00071 1
00071 1 <<OTHER INITIALIZATIONS>>
00071 1 FAILURE := FALSE;
00073 1 ERROR := TRUE;
00075 1
00075 1 <<PERFORM THE MERGE AND CHECK FOR ERRORS>>
00075 1
00075 1 MERGEINIT(INPUT,,OUTPUT,,,1,KEYS,,,,STATISTICS,FAILURE,ERROR
00112 1 MERGEEND;

```

PAGE 0002 HEWLETT-PACKARD

```

00113 1
00113 1 IF FAILURE
00113 1 THEN BEGIN << PRINT THE ERROR MESSAGE AND NUMBER >>
00115 2 MERGEERRORMESS(ERROR,BUFFER,LENGTH);
00121 2 PRINT(WORD'BUF,-LENGTH,%320);
00126 2 MOVE BUFFER := `` ('';
00143 2 LENGTH := ASCII(ERROR,10,BUFFER(3)) + 3;
00153 2 MOVE BUFFER(LENGTH) := ``)'';
00163 2 LENGTH := LENGTH + 2;
00166 2 PRINT(WORD'BUF,-LENGTH, %40);
00173 2 END
00173 1 ELSE << PRINT THE STATISTICS >>
00174 1 MERGESTAT(STATISTICS);
00176 1
00176 1 FCLOSE(INPUT(0),NO'CHANGE,UNRESTRICTED);
00202 1 FCLOSE(INPUT(1),NO'CHANGE,UNRESTRICTED);
00206 1 FCLOSE(INPUT(2),NO'CHANGE,UNRESTRICTED);
00212 1
00212 1 FCLOSE(OUTPUT(0),SAVE'PERM+RETURN'SPACE,UNRESTRICTED);
00220 1
00220 1 END.
PRIMARY DB STORAGE=%016; SECONDARY DB STORAGE=%00122
NO. ERRORS=0000; NO. WARNINGS=0000
PROCESSOR TIME=0:00:02; ELAPSED TIME=0:00:06

```

END OF COMPILE  
:PREP \$OLDPASS,PNOBUF4;MAXDATA=31232

END OF PREPARE  
:RUN PNOBUF4;LIB=G

#### STATISTICS

```

NUMBER OF INPUT FILES = 3
NUMBER OF RECORDS = 150
SPACE AVAILABLE (IN WORDS) = 28,097
NUMBER OF COMPARES = 243
CPU TIME (MINUTES) = .02
ELAPSED TIME (MINUTES) = .10

```

END OF PROGRAM

## Calling MERGE with the *charseq* parameter.

```
00000 0 $CONTROL USLINIT
00000 0 << SPL EXAMPLE M1 >>
00000 0 << SPL CHARSEQ PARAMETER EXAMPLE >>
00000 0 << SORT THE FILES, SWED1 AND SWED2,, INTO THE FILE, SWED3 >>
00000 0 << SORT ON THE FIRST 5 CHARACTERS USING THE COLLATING SEQUENCE
00000 0 FOR THE NATIVE LANGUAGE, SWEDISH >>
00000 0
00000 0 BEGIN
00006 1 BYTE ARRAY INPUT1(0:9) := "SWED1 ";
00006 1 BYTE ARRAY INPUT2(0:9) := "SWED2 ";
00006 1 BYTE ARRAY OUTPUT(0:9) := "SWED3 ";
00006 1 ARRAY BUF(0:21);
00006 1 INTEGER ARRAY IN(0:2), OUT(0:1);
00006 1 INTEGER LEN;
00006 1 INTEGER NUMKEYS := 1;
00006 1 INTEGER ARRAY KEYS(0:2);
00006 1 INTEGER ARRAY CSEQ(0:1);
00006 1 INTRINSIC FPOINT, QUIT, FOPEN, FREAD, PRIN, MERGEINIT,
00006 1 MERGEEND;
00006 1
00006 1 IN(0) := FOPEN(INPUT1,5);
00010 1 IN(1) := FOPEN(INPUT2,5);
00020 1 IN(2) := 0;
00023 1 IF <> THEN QUIT(90);
00026 1 OUT(0) := FOPEN(OUTPUT,4,4);
00037 1 OUT(1) := 0;
00042 1 IF <> THEN QUIT(91);
00045 1 << ESTABLISH THE KEY WITH KEY TYPE CHARACTER >>
00045 1 KEYS(0) := 1;
00050 1 KEYS(1) := 5;
00053 1 KEYS(2) := 9;
00056 1 << ESTABLISH THE CHARSEQ PARAMETER; LANGUAGE = SWEDISH
00056 1 LANGNUM = 13 >>
00056 1 CSEQ(0) := 1;
00061 1 CSEQ(1) := 13;
00064 1 << CALL MERGEINIT WITH THE CHARSEQ PATAMETER >>
00064 1 MERGEINIT(IN,,OUT,,NUMKEYS,KEYS,,,,,CSEQ);
00077 1 MERGEEND;
00100 1 FPOINT(OUT,00);
00103 1 LOOP : LEN := FREAD(OUT,BUF,21);
00111 1 IF <> THEN QUIT(93);
00114 1 PRINT(BUF,LEN,0);
00120 1 GOTO LOOP;
00124 1 END.
PRIMARY OR STORAGE=%:12; SECONDARY OB STORAGE=%00057
NO.ERRORS=0000; NO. WARNINGS=0000
PROCESSOR TIME=0:00:00; ELAPSED TIME=0:00:07

END OF COMPILE
:BUILD swed3;rec=-80,,f,ascii
:PRERUN $oldpass;maxdata=30000

END OF PREPARE
```

Anderson, Hans  
Bergman, Ingrid  
Bergman, Ingmar  
Borg, Bjorn  
Hammerskjold, Dag  
Svenson, Bo  
Ullman, Liv



# ERROR MESSAGES AND RECOVERY PROCEDURES

APPENDIX

A

## SORT ERROR MESSAGES

The Table A-1 contains messages issued by the SORTLIB segment of the system segmented library. The messages marked by I/O in the second column of the table result in a file information display. The messages marked by SA in the second column can be issued by the SORT standalone program. The remaining messages, marked by LIB in the second column, are not issued by the stand-alone SORT program but are displayed when SORT is performed programmatically and an error occurs. Each message consists of SORTLIB: followed by the text of the message.

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                  |
|--------------|---------------|----------------------------------------------------------|
| 1            | LIB           | IF KEYCOMPARE IS SPECIFIED, KEYS AND NUMKEYS MUST NOT BE |
| 2            | LIB           | IF KEYCOMPARE IS NOT SPECIFIED, KEYS AND NUMKEYS MUST BE |
| 3            | LIB           | NO RECLEN PARAMETER SPECIFIED OR <=0                     |
| 4            | LIB           | KEYCOMPARE MAY NOT BE SPECIFIED IF OUTPUTOPTION > 1      |
| 5            | I/O           | FREAD ERROR ON SCRATCH FILE                              |
| 6            | LIB           | ILLEGAL OUTPUTOPTION                                     |
| 7            | I/O           | SCRATCH FILE CANNOT BE OPENED                            |
| 8            | LIB, I/O      | FAILURE ON FGETINFO(INPUTFILE)                           |
| 9            | LIB           | ILLEGAL NUMKEYS                                          |
| 10           | SA            | KEYFIELD IS NOT WITHIN SPECIFIED RECORD LENGTH           |
| 11           | LIB           | ILLEGAL ASCENDING/DESCENDING CODE                        |
| 12           | LIB           | ILLEGAL KEY CODE                                         |
| 13           | SA            | INSUFFICIENT STACK SPACE                                 |
| 14           | SA            | INPUT RECORD DOES NOT INCLUDE ALL KEY FIELDS             |
| 15           | LIB           | INPUT RECORD IS TOO LONG                                 |
| 16           | SA            | TOO MANY INPUT RECORDS                                   |
| 17           | I/O           | FWRITE ERROR ON SCRATCH FILE                             |
| 18           | I/O           | FREAD ERROR ON INPUT FILE                                |
| 19           | I/O           | FWRITE ERROR ON OUTPUT FILE                              |
| 20           | I/O           | FCLOSE ERROR ON SCRATCH FILE                             |
| 21           | I/O           | \$NULL IS NOT A VALID INPUT FILE                         |



| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                   |
|--------------|---------------|-----------------------------------------------------------|
| 23           | I/O           | ERROR ATTEMPTING TO WRITE EOF ON SCRATCH FILE             |
| 24           | I/O           | ERROR ATTEMPTING TO REWIND SCRATCH FILE                   |
| 25           | I/O           | ILLEGAL CHARACTERISTIC FOR FOPEN OF SCRATCH FILE          |
| 26           | LIB           | INSUFFICIENT STACK SPACE FOR SPECIFIED ALLOCATION         |
| 29           | LIB           | SORT LANGUAGE NOT SUPPORTED                               |
| 30           | LIB           | NLINFO ERROR OBTAINING LENGTH OF COLLATING SEQUENCE TABLE |
| 31           | LIB           | NLINFO ERROR LOADING COLLATING SEQUENCE TABLE             |
| 32           | LIB           | INVALID CHARSEQ PARAMETER                                 |

Table A-1. SORTLIB Error Messages

Table A-2 contains messages issued by the SORT standalone program. The messages containing I/O in the second column result in a file information display. Those marked with HARD in the second column terminate the program. All others also cause program termination, unless the program is run interactively, in which case you are asked to enter the command again. The stand-alone SORT program commands listed in the fourth column of the table are the commands that immediately precede errors during SORT.

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                  | COMMAND                                |
|--------------|---------------|------------------------------------------|----------------------------------------|
| 1            | I/O, HARD     | FAILURE ON FOPEN OF LIST FILE            |                                        |
| 2            | HARD          | LIST FILE IS READ-ONLY                   |                                        |
| 3            | I/O, HARD     | FAILURE ON FOPEN OF TEXT FILE            |                                        |
| 4            | HARD          | TEXT FILE IS WRITE-ONLY                  |                                        |
| 5            |               | ILLEGAL COMMAND                          |                                        |
| 6            |               | NO KEYS WERE SPECIFIED                   | END                                    |
| 7            |               | FILENAME CANNOT EXCEED 35 CHARACTERS     | INPUT, OUTPUT                          |
| 8            |               | MISSING COMMA                            | INPUT, OUTPUT, KEY                     |
| 9            |               | ILLEGAL NUMBER OF RECORDS                | INPUT                                  |
| 10           |               | NUMBER OF RECORDS TOO LARGE OR TOO SMALL | INPUT                                  |
| 11           |               | ILLEGAL RECORD SIZE                      | INPUT                                  |
| 12           |               | RECORD SIZE TOO LARGE OR TOO SMALL       | INPUT                                  |
| 13           |               | TOO MANY PARAMETERS                      | INPUT, OUTPUT, KEY, RESET, VERIFY, END |
| 14           | I/O           | FAILURE ON FOPEN OF INPUT FILE           | END                                    |
| 15           |               | MISSING NUM OR KEY                       | OUTPUT                                 |

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                              | COMMAND            |
|--------------|---------------|----------------------------------------------------------------------|--------------------|
| 16           |               | ILLEGAL POSITION                                                     | KEY                |
| 17           |               | POSITION OUT OF RANGE                                                | KEY                |
| 18           |               | MISSING PARAMETER                                                    | INPUT, OUTPUT, KEY |
| 19           |               | LENGTH OUT OF RANGE                                                  | KEY                |
| 20           |               | LENGTH PARAMETER NOT AN INTEGER                                      | KEY                |
| 21           |               | LENGTH NOT SPECIFIED FOR TYPE BYTE, PACKED, OR DISPLAY               | KEY                |
| 22           |               | MISSING DESC                                                         |                    |
| 23           |               | INPUT FILE IS WRITE-ONLY                                             | END                |
| 24           |               | FAILURE ON FOPEN OF OUTPUT FILE                                      | END                |
| 25           |               | OUTPUT FILE IS READ-ONLY                                             | END                |
| 26           | I/O           | FAILURE ON FCLOSE OF OUTPUT FILE                                     |                    |
| 27           |               | SUM OF KEYFIELDS SIZES TOO LARGE                                     |                    |
| 28           | I/O           | FAILURE ON PURGE OF OLD OUTPUT FILE                                  |                    |
| 29           | I/O           | FAILURE ON FOPEN OF OLD OUTPUT FILE                                  |                    |
| 30           | I/O           | FAILURE ON FRENAME OF OUTPUT FILE                                    |                    |
| 31           | I/O, HARD     | FAILURE ON FWRITE OF PROMPT FILE                                     |                    |
| 32           | I/O, HARD     | FAILURE ON FREAD OF TEXT FILE                                        |                    |
| 33           |               | INSUFFICIENT STACK SPACE                                             |                    |
| 34           |               | MISSING PARAMETER                                                    |                    |
| 35           |               | ERROR, SYNTAX IS: DATA [IS] ASCII/EBCDIC, SEQUENCE [IS] ASCII/EBCDIC |                    |
| 36           |               | ERROR, SYNTAX IS: SHOW [NO] SEQUENCE/ [NO] TABLE [, OFFLINE]         |                    |
| 37           |               | A USER DEFINED SEQUENCE CAN ONLY BE SPECIFIED WHEN DATA IS ASCII     |                    |
| 38           |               | THE DATA COMMAND MUST BE ISSUED BEFORE THE ALTSEQ OR SHOW COMMANDS   |                    |
| 39           | I/O           | \$NULL IS NOT A VALID INPUT FILE                                     |                    |
| 40           |               | INVALID LANGUAGE ID                                                  | LANGUAGE           |
| 41           |               | THE LANGUAGE SPECIFIED IS NOT SUPPORTED                              | LANGUAGE           |

Table A-2. SORT Program Error Messages

## ALTSEQ ERROR MESSAGES

Table A-3 lists the messages issued if the ALTSEQ command is incorrectly specified. Recovery from these errors is accomplished by reentering the command during interactive sessions.

| ERROR NUMBER | MESSAGE                                                                     |
|--------------|-----------------------------------------------------------------------------|
| 2            | INVALID DIGIT FOR BASE SPECIFIED                                            |
| 3            | INVALID PARAMETER                                                           |
| 4            | INVALID COMMAND, SYNTAX IS: ALTSEQ [EACH/MERGE]<br>modspec1 =/WITH modspec2 |
| 5            | THE STRING MUST BE CONTINUOUS AND STRICTLY INCREASING                       |
| 6            | AN INVALID CHARACTER FOUND IN BYTE SPECIFICATION                            |
| 11           | THE BASE IS OUT OF THE RANGE 2 THRU 16                                      |
| 12           | THE LENGTH OF THE SPEC STRING CANNOT BE ZERO                                |
| 13           | "EACH" DOESN'T MAKE SENSE IN THIS CONTEXT                                   |
| 14           | "MERGE" DOESN'T MAKE SENSE IN THIS CONTEXT                                  |
| 15           | INVALID RANGE SPECIFICATION                                                 |
| 16           | MERGE STRINGS MAY NOT OVERLAP                                               |
| 17           | A BYTE SPECIFICATION IS GREATER THAN 255. PLEASE RESPECIFY                  |

Table A-3 ALTSEQ Error Messages

## MERGE ERROR MESSAGES

Table A-4 lists the messages issued by the MERGELIB segment of the system segmented library. The messages containing I/O in the second column result in a file information display. The remaining messages are not displayed by the stand-alone MERGE program; but are printed (if an error occurs), if MERGE is performed programmatically. Each message consists of MERGELIB: followed by the text of the message.

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                   |
|--------------|---------------|-----------------------------------------------------------|
| 1            | LIB           | NO NUMINPUTFILES PARAMETER SPECIFIED                      |
| 2            | LIB           | ILLEGAL NUMINPUTFILES                                     |
| 3            | LIB           | NO INPUTFILES PARAMETER SPECIFIED                         |
| 4            | LIB           | NEITHER OUTPUTFILE NOR POSTPROCESSOR PARAMETER SPECIFIED  |
| 5            | LIB           | IF KEYCOMPARE IS SPECIFIED, KEYS AND NUMKEYS MUST NOT BE  |
| 6            | LIB           | IF KEYCOMPARE IS NOT SPECIFIED, KEYS AND NUMKEYS MUST BE  |
| 7            | LIB           | ILLEGAL NUMKEYS                                           |
| 8            | LIB           | KEYFIELD IS NOT WITHIN RECORD LENGTH OF EACH FILE         |
| 9            | LIB           | ILLEGAL ASCENDING/DESCENDING CODE                         |
| 10           | LIB           | ILLEGAL KEY CODE                                          |
| 11           | LIB, I/O      | FAILURE ON FGETINFO(INPUTFILE)                            |
| 12           | I/O           | FREAD ERROR ON INPUT FILE                                 |
| 13           | I/O           | FWRITE ERROR ON OUTPUT FILE                               |
| 14           | I/O           | INPUT RECORD DOES NOT INCLUDE ALL KEY FIELDS              |
| 15           | LIB           | IF KEYCOMPARE IS SPECIFIED, KEYONLY MAY NOT BE            |
| 16           |               | INSUFFICIENT STACK SPACE                                  |
| 17           | LIB           | INSUFFICIENT STACK SPACE FOR SPECIFIED ALLOCATION         |
| 18           | I/O           | FAILURE ON FGETINFO (OUTPUTFILE)                          |
| 19           | I/O           | \$NULL IS NOT A VALID INPUT FILE                          |
| 21           | LIB           | SORT LANGUAGE NOT SUPPORTED                               |
| 22           | LIB           | NLINFO ERROR OBTAINING LENGTH OF COLLATING SEQUENCE TABLE |
| 23           | LIB           | NLINFO ERROR LOADING COLLATING SEQUENCE TABLE             |
| 24           | LIB           | INVALID CHARSEQ PARAMETER                                 |

Table A-4. MERGELIB Error Messages

Table A-5 lists the messages issued by the MERGE standalone program. The messages containing I/O in the second column result in a file information display. Those marked with HARD in the second column terminate the program. All others also cause program termination unless the program is run interactively, in which case you are asked to enter the command again. The stand-alone MERGE program commands listed in the fourth column cause errors during MERGE.

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                | COMMAND                         |
|--------------|---------------|--------------------------------------------------------|---------------------------------|
| 1            | I/O, HARD     | FAILURE ON FOPEN OF LIST FILE                          |                                 |
| 2            | HARD          | LIST FILE IS READ-ONLY                                 |                                 |
| 3            | I/O, HARD     | FAILURE ON FOPEN OF TEXT FILE                          |                                 |
| 4            | HARD          | TEXT FILE IS WRITE-ONLY                                |                                 |
| 5            |               | ILLEGAL COMMAND                                        |                                 |
| 6            |               | NO KEYS WERE SPECIFIED                                 | END                             |
| 7            |               | FILENAME CANNOT EXCEED 35 CHARACTERS                   | INPUT, OUTPUT                   |
| 8            |               | MISSING COMMA                                          | INPUT, OUTPUT, KEY              |
| 9            |               | MISSING PARAMETER                                      | INPUT, OUTPUT, KEY              |
| 10           |               | ILLEGAL NUMBER OF RECORDS                              | OUTPUT                          |
| 11           |               | NUMBER OF RECORDS TOO LARGE OR TOO SMALL               | OUTPUT                          |
| 12           |               | TOO MANY PARAMETERS                                    | OUTPUT, KEY, RESET, VERIFY, END |
| 13           | HARD          | INSUFFICIENT SPACE                                     |                                 |
| 14           | I/O, HARD     | FAILURE ON FOPEN OF INPUT FILE                         | END                             |
| 15           |               | ILLEGAL POSITION                                       | KEY                             |
| 16           |               | POSITION OUT OF RANGE                                  | KEY                             |
| 17           |               | LENGTH OUT OF RANGE                                    | KEY                             |
| 18           |               | LENGTH PARAMETER NOT AN INTEGER                        | KEY                             |
| 19           |               | LENGTH NOT SPECIFIED FOR TYPE BYTE, PACKED, OR DISPLAY | KEY                             |
| 20           |               | MISSING DESC                                           | KEY                             |
| 21           | I/O, HARD     | INPUT FILE IS WRITE-ONLY                               | KEY                             |
| 22           | I/O, HARD     | FAILURE ON FOPEN OF OUTPUT FILE                        | END                             |
| 23           | I/O, HARD     | OUTPUT FILE IS READ-ONLY                               | END                             |
| 24           |               | NO INPUT FILES WERE SPECIFIED                          | END                             |
| 25           |               | FAILURE ON FCLOSE OF OUTPUT FILE                       |                                 |

| ERROR NUMBER | TYPE OF ERROR | MESSAGE                                                               | COMMAND  |
|--------------|---------------|-----------------------------------------------------------------------|----------|
| 26           |               | SUM OF KEYFIELD SIZES TOO LARGE                                       |          |
| 27           | I/O           | FAILURE ON PURGE OF OLD OUTPUT FILE                                   |          |
| 28           | I/O           | FAILURE ON FOPEN OF OLD OUTPUT FILE                                   |          |
| 29           | I/O           | FAILURE ON FRENAME OF OUTPUT FILE                                     |          |
| 30           | I/O, HARD     | FAILURE ON FWRITE OF PROMPT FILE                                      |          |
| 31           | I/O, HARD     | FAILURE ON FREAD OF TEXT FILE                                         |          |
| 32           |               | ERROR, SYNTAX IS: DATA [IS]<br>ASCII/EBCDIC                           |          |
| 33           |               | ERROR, SYNTAX IS: SHOW [NO]<br>SEQUENCE/[NO] TABLE [, OFFLINE]        |          |
| 34           |               | A USER DEFINED SEQUENCE CAN ONLY<br>BE SPECIFIED WHEN DATA IS ASCII   |          |
| 35           |               | THE DATA COMMAND MUST BE ISSUED<br>BEFORE THE ALTSEQ OR SHOW COMMANDS |          |
| 36           | I/O           | \$NULL IS NOT A VALID INPUT FILE                                      |          |
| 37           |               | INVALID LANGUAGE ID                                                   | LANGUAGE |
| 38           |               | THE LANGUAGE SPECIFIED IS NOT<br>SUPPORTED                            | LANGUAGE |

Table A-5. MERGE Program Error Messages

## RECOVERY PROCEDURES

If you wish your program to continue when SORTLIB or MERGELIB errors occur, you must call the SORTEND or MERGEEND intrinsic in order to restore the stack to its original condition. The remainder of your program continues to run. When an error occurs in the MERGELIB procedure, no recovery is necessary since the procedure returns directly to your program. Errors during stand-alone SORT (or MERGE) in the batch mode are not recoverable and the programs terminate abnormally. In interactive sessions, syntax errors are recoverable and you are asked to enter the command again.



# ASCII/EBCDIC/HOLLERITH TABLE

APPENDIX

B

The table is sorted by the character code value, each value represented by its decimal, octal, and hexadecimal equivalent. Each row of the table gives the ASCII and EBCDIC code values of the particular character, the ASCII-EBCDIC conversion, and the Hollerith representation (punched card code value) for the ASCII character.

## EXAMPLES

If you want to determine the ASCII code value of the character \$, scan down the ASCII graphic column until you locate \$. Then read to its left to find the values 36 (decimal), 044 (octal), or 24 (hexadecimal). This is the code value used by devices such as terminal, printer, cpu, etc, to represent the character \$. Its Hollerith code value is 11-3-8.

To find the character whose EBCDIC code value is 5B (hexadecimal), locate 5B in the Hexadecimal character value column and move right to the EBCDIC graphic column which gives \$. The next column to the right of \$ gives the conversion to the ASCII code value—that is, 044 (octal). As a check, locate 044 in the octal value column. Then look to the right of the ASCII graphic column. Note \$ is converted to EBCDIC 133 (octal) which equals 5B (hexadecimal).

| CHAR CODE |     |     | ASCII        |                       |             | EBCDIC       |                      |  |
|-----------|-----|-----|--------------|-----------------------|-------------|--------------|----------------------|--|
| Dec       | Oct | Hex | Cntl/<br>Gph | to<br>EBCDIC<br>(Oct) | Hollerith   | Cntl/<br>Gph | to<br>ASCII<br>(Oct) |  |
| 0         | 000 | 00  | NUL          | 000                   | 12-0-1-8-9  | NUL          | 000                  |  |
| 1         | 001 | 01  | SOH          | 001                   | 12-1-9      | SOH          | 001                  |  |
| 2         | 002 | 02  | STX          | 002                   | 12-2-9      | STX          | 002                  |  |
| 3         | 003 | 03  | ETX          | 003                   | 12-3-9      | ETX          | 003                  |  |
| 4         | 004 | 04  | EOT          | 067                   | 7-9         | PF           | 234                  |  |
| 5         | 005 | 05  | ENQ          | 055                   | 0-5-8-9     | HT           | 011                  |  |
| 6         | 006 | 06  | ACK          | 056                   | 0-6-8-9     | LC           | 206                  |  |
| 7         | 007 | 07  | BEL          | 057                   | 0-7-8-9     | DEL          | 177                  |  |
| 8         | 010 | 08  | BS           | 026                   | 11-6-9      |              | 227                  |  |
| 9         | 011 | 09  | HT           | 005                   | 12-5-9      |              | 215                  |  |
| 10        | 012 | 0A  | LF           | 045                   | 0-5-9       | SMM          | 216                  |  |
| 11        | 013 | 0B  | VT           | 013                   | 12-3-8-9    | VT           | 013                  |  |
| 12        | 014 | 0C  | FF           | 014                   | 12-4-8-9    | FF           | 014                  |  |
| 13        | 015 | 0D  | CR           | 015                   | 12-5-8-9    | CR           | 015                  |  |
| 14        | 016 | 0E  | SO           | 016                   | 12-6-8-9    | SO           | 016                  |  |
| 15        | 017 | 0F  | SI           | 017                   | 12-7-8-9    | SI           | 017                  |  |
| 16        | 020 | 10  | OLE          | 020                   | 12-11-1-8-9 | DLE          | 020                  |  |
| 17        | 021 | 11  | DC1          | 021                   | 11-1-9      | DC1          | 021                  |  |
| 18        | 022 | 12  | DC2          | 022                   | 11-2-9      | DC2          | 022                  |  |
| 19        | 023 | 13  | DC3          | 023                   | 11-3-9      | TM           | 023                  |  |
| 20        | 024 | 14  | DC4          | 074                   | 4-8-9       | RES          | 235                  |  |
| 21        | 025 | 15  | NAK          | 075                   | 5-8-9       | NL           | 205                  |  |
| 22        | 026 | 16  | SYN          | 062                   | 2-9         | BS           | 010                  |  |
| 23        | 027 | 17  | ETB          | 046                   | 0-6-9       | IL           | 207                  |  |
| 24        | 030 | 18  | CAN          | 030                   | 11-8-9      | CAN          | 030                  |  |
| 25        | 031 | 19  | EM           | 031                   | 11-1-8-9    | EM           | 031                  |  |
| 26        | 032 | 1A  | SUB          | 077                   | 7-8-9       | CC           | 222                  |  |
| 27        | 033 | 1B  | ESC          | 047                   | 0-7-9       | CU1          | 217                  |  |
| 28        | 034 | 1C  | FS           | 034                   | 11-4-8-9    | IFS          | 034                  |  |
| 29        | 035 | 1D  | GS           | 035                   | 11-5-8-9    | IGS          | 035                  |  |
| 30        | 036 | 1E  | RS           | 036                   | 11-6-8-9    | IRS          | 036                  |  |
| 31        | 037 | 1F  | US           | 037                   | 11-7-8-9    | IUS          | 037                  |  |
| 32        | 040 | 20  | SP           | 100                   | Blank       | DS           | 200                  |  |
| 33        | 041 | 21  | '            | 117                   | 12-7-8      | SOS          | 201                  |  |
| 34        | 042 | 22  | "            | 177                   | 7-8         | FS           | 202                  |  |
| 35        | 043 | 23  | #            | 173                   | 3-8         |              | 203                  |  |
| 36        | 044 | 24  | \$           | 133                   | 11-3-8      | 8YP          | 204                  |  |
| 37        | 045 | 25  | %            | 154                   | 0-4-8       | LF           | 012                  |  |
| 38        | 046 | 26  | &            | 120                   | 12          | ETB          | 027                  |  |
| 39        | 047 | 27  | '            | 175                   | 5-8         | ESC          | 033                  |  |
| 40        | 050 | 28  | (            | 115                   | 12-5-8      |              | 210                  |  |
| 41        | 051 | 29  | )            | 135                   | 11-5-8      |              | 211                  |  |
| 42        | 052 | 2A  | *            | 134                   | 11-4-8      | SM           | 212                  |  |
| 43        | 053 | 2B  | +            | 116                   | 12-6-8      | CU2          | 213                  |  |
| 44        | 054 | 2C  | ,            | 153                   | 0-3-8       |              | 214                  |  |
| 45        | 055 | 2D  | -            | 140                   | 11          | ENQ          | 005                  |  |
| 46        | 056 | 2E  | .            | 113                   | 12-3-8      | ACK          | 006                  |  |
| 47        | 057 | 2F  | /            | 141                   | 0-1         | BEL          | 007                  |  |

| CHAR CODE |     |     | ASCII        |                       |           | EBCDIC       |                      |  |
|-----------|-----|-----|--------------|-----------------------|-----------|--------------|----------------------|--|
| Dec       | Oct | Hex | Cntl/<br>Gph | to<br>EBCDIC<br>(Oct) | Hollerith | Cntl/<br>Gph | to<br>ASCII<br>(Oct) |  |
| 48        | 060 | 30  |              | 360                   | 0         |              | 220                  |  |
| 49        | 061 | 31  |              | 361                   | 1         |              | 221                  |  |
| 50        | 062 | 32  |              | 362                   | 2         | SYN          | 026                  |  |
| 51        | 063 | 33  |              | 363                   | 3         |              | 223                  |  |
| 52        | 064 | 34  |              | 364                   | 4         | PN           | 224                  |  |
| 53        | 065 | 35  |              | 365                   | 5         | RS           | 225                  |  |
| 54        | 066 | 36  |              | 366                   | 6         | UC           | 226                  |  |
| 55        | 067 | 37  |              | 367                   | 7         | EOT          | 004                  |  |
| 56        | 070 | 38  |              | 370                   | 8         |              | 230                  |  |
| 57        | 071 | 39  |              | 371                   | 9         |              | 231                  |  |
| 58        | 072 | 3A  |              | 172                   | 2-8       |              | 232                  |  |
| 59        | 073 | 3B  |              | 136                   | 11-8-8    | CU3          | 233                  |  |
| 60        | 074 | 3C  | <            | 114                   | 12-4-8    | DC4          | 024                  |  |
| 61        | 075 | 3D  | =            | 176                   | 6-8       | NAK          | 025                  |  |
| 62        | 076 | 3E  | >            | 156                   | 0-6-8     |              | 236                  |  |
| 63        | 077 | 3F  | ?            | 157                   | 0-7-8     | SUB          | 032                  |  |
| 64        | 100 | 40  | @            | 174                   | 4-8       | SP           | 040                  |  |
| 65        | 101 | 41  | A            | 301                   | 12-1      |              | 240                  |  |
| 66        | 102 | 42  | B            | 302                   | 12-2      |              | 241                  |  |
| 67        | 103 | 43  | C            | 303                   | 12-3      |              | 242                  |  |
| 68        | 104 | 44  | D            | 304                   | 12-4      |              | 243                  |  |
| 69        | 105 | 45  | E            | 305                   | 12-5      |              | 244                  |  |
| 70        | 106 | 46  | F            | 306                   | 12-6      |              | 245                  |  |
| 71        | 107 | 47  | G            | 307                   | 12-7      |              | 246                  |  |
| 72        | 110 | 48  | H            | 310                   | 12-8      |              | 247                  |  |
| 73        | 111 | 49  | I            | 311                   | 12-9      |              | 250                  |  |
| 74        | 112 | 4A  | J            | 321                   | 11-1      |              | 133                  |  |
| 75        | 113 | 4B  | K            | 322                   | 11-2      |              | 056                  |  |
| 76        | 114 | 4C  | L            | 323                   | 11-3      | <            | 074                  |  |
| 77        | 115 | 4D  | M            | 324                   | 11-4      | (            | 050                  |  |
| 78        | 116 | 4E  | N            | 325                   | 11-5      | +            | 053                  |  |
| 79        | 117 | 4F  | O            | 326                   | 11-6      | )            | 041                  |  |
| 80        | 120 | 50  | P            | 327                   | 11-7      | &            | 046                  |  |
| 81        | 121 | 51  | Q            | 330                   | 11-8      |              | 251                  |  |
| 82        | 122 | 52  | R            | 331                   | 11-9      |              | 252                  |  |
| 83        | 123 | 53  | S            | 342                   | 0-2       |              | 253                  |  |
| 84        | 124 | 54  | T            | 343                   | 0-3       |              | 254                  |  |
| 85        | 125 | 55  | U            | 344                   | 0-4       |              | 255                  |  |
| 86        | 126 | 56  | V            | 345                   | 0-5       |              | 256                  |  |
| 87        | 127 | 57  | W            | 346                   | 0-6       |              | 257                  |  |
| 88        | 130 | 58  | X            | 347                   | 0-7       |              | 260                  |  |
| 89        | 131 | 59  | Y            | 350                   | 0-8       |              | 261                  |  |
| 90        | 132 | 5A  | Z            | 351                   | 0-9       | !            | 135                  |  |
| 91        | 133 | 5B  | [            | 112                   | 12-2-8    | \$           | 044                  |  |
| 92        | 134 | 5C  | \            | 340                   | 0-2-8     | *            | 052                  |  |
| 93        | 135 | 5D  | ]            | 132                   | 11-2-8    | )            | 051                  |  |
| 94        | 136 | 5E  | ^            | 137                   | 11-7-8    | ;            | 073                  |  |
| 95        | 137 | 5F  | _            | 155                   | 0-5-8     | ~            | 136                  |  |



| CHAR CODE |     |     | ASCII        |                       |               | EBCDIC       |                      |
|-----------|-----|-----|--------------|-----------------------|---------------|--------------|----------------------|
| Dec       | Oct | Hex | Cntl/<br>Gph | to<br>EBCDIC<br>(Oct) | Hollerith     | Cntl/<br>Gph | to<br>ASCII<br>(Oct) |
| 96        | 140 | 60  |              | 171                   | 1-8           | -            | 055                  |
| 97        | 141 | 61  | a            | 201                   | 12-0-1        | /            | 057                  |
| 98        | 142 | 62  | b            | 202                   | 12-0-2        |              | 262                  |
| 99        | 143 | 63  | c            | 203                   | 12-0-3        |              | 263                  |
| 100       | 144 | 64  | d            | 204                   | 12-0-4        |              | 264                  |
| 101       | 145 | 65  | e            | 205                   | 12-0-5        |              | 265                  |
| 102       | 146 | 66  | f            | 206                   | 12-0-6        |              | 266                  |
| 103       | 147 | 67  | g            | 207                   | 12-0-7        |              | 267                  |
| 104       | 150 | 68  | h            | 210                   | 12-0-8        |              | 270                  |
| 105       | 151 | 69  | i            | 211                   | 12-0-9        |              | 271                  |
| 106       | 152 | 6A  | j            | 221                   | 12-11-1       |              | 174                  |
| 107       | 153 | 6B  | k            | 222                   | 12-11-2       |              | 054                  |
| 108       | 154 | 6C  | l            | 223                   | 12-11-3       | %            | 045                  |
| 109       | 155 | 6D  | m            | 224                   | 12-11-4       |              | 137                  |
| 110       | 156 | 6E  | n            | 225                   | 12-11-5       | ∨            | 076                  |
| 111       | 157 | 6F  | o            | 226                   | 12-11-6       | ?            | 077                  |
| 112       | 160 | 70  | p            | 227                   | 12-11-7       |              | 272                  |
| 113       | 161 | 71  | q            | 230                   | 12-11-8       |              | 273                  |
| 114       | 162 | 72  | r            | 231                   | 12-11-9       |              | 274                  |
| 115       | 163 | 73  | s            | 242                   | 11-0-2        |              | 275                  |
| 116       | 164 | 74  | t            | 243                   | 11-0-3        |              | 276                  |
| 117       | 165 | 75  | u            | 244                   | 11-0-4        |              | 277                  |
| 118       | 166 | 76  | v            | 245                   | 11-0-5        |              | 300                  |
| 119       | 167 | 77  | w            | 246                   | 11-0-6        |              | 301                  |
| 120       | 170 | 78  | x            | 247                   | 11-0-7        |              | 302                  |
| 121       | 171 | 79  | y            | 250                   | 11-0-8        |              | 140                  |
| 122       | 172 | 7A  | z            | 251                   | 11-0-9        |              | 072                  |
| 123       | 173 | 7B  | {            | 300                   | 12-0          | U            | 043                  |
| 124       | 174 | 7C  |              | 152                   | 12-11         |              | 100                  |
| 125       | 175 | 7D  | }            | 320                   | 11-0          | @            | 047                  |
| 126       | 176 | 7E  | DEL          | 241                   | 11-0-1        |              | 075                  |
| 127       | 177 | 7F  |              | 007                   | 12-7-9        |              | 042                  |
| 128       | 200 | 80  |              | 040                   | 11-0-1-8-9    |              | 303                  |
| 129       | 201 | 81  | a            | 041                   | 0-1-9         | a            | 141                  |
| 130       | 202 | 82  | b            | 042                   | 0-2-9         | b            | 142                  |
| 131       | 203 | 83  | c            | 043                   | 0-3-9         | c            | 143                  |
| 132       | 204 | 84  | d            | 044                   | 0-4-9         | d            | 144                  |
| 133       | 205 | 85  | e            | 025                   | 11-5-9        | e            | 145                  |
| 134       | 206 | 86  | f            | 006                   | 12-6-9        | f            | 146                  |
| 135       | 207 | 87  | g            | 027                   | 11-7-9        | g            | 147                  |
| 136       | 210 | 88  | h            | 050                   | 0-8-9         | h            | 150                  |
| 137       | 211 | 89  | i            | 051                   | 0-1-8-9       | i            | 151                  |
| 138       | 212 | 8A  |              | 052                   | 0-2-8-9       |              | 304                  |
| 139       | 213 | 8B  |              | 053                   | 0-3-8-9       |              | 305                  |
| 140       | 214 | 8C  |              | 054                   | 0-4-8-9       |              | 306                  |
| 141       | 215 | 8D  |              | 011                   | 12-1-8-9      |              | 307                  |
| 142       | 216 | 8E  |              | 012                   | 12-2-8-9      |              | 310                  |
| 143       | 217 | 8F  |              | 033                   | 11-3-8-9      |              | 311                  |
| 144       | 220 | 90  |              | 060                   | 12-11-0-1-8-9 |              | 312                  |
| 145       | 221 | 91  | j            | 061                   | 1-9           | j            | 152                  |
| 146       | 222 | 92  | k            | 032                   | 11-2-8-9      | k            | 153                  |
| 147       | 223 | 93  | l            | 063                   | 3-9           | l            | 154                  |
| 148       | 224 | 94  | m            | 064                   | 4-9           | m            | 155                  |
| 149       | 225 | 95  | n            | 065                   | 5-9           | n            | 156                  |
| 150       | 226 | 96  | o            | 066                   | 6-9           | o            | 157                  |
| 151       | 227 | 97  | p            | 010                   | 12-8-9        | p            | 160                  |
| 152       | 230 | 98  | q            | 070                   | 8-9           | q            | 161                  |
| 153       | 231 | 99  | r            | 071                   | 1-8-9         | r            | 162                  |
| 154       | 232 | 9A  |              | 072                   | 2-8-9         |              | 313                  |
| 155       | 233 | 9B  |              | 073                   | 3-8-9         |              | 314                  |
| 156       | 234 | 9C  |              | 004                   | 12-4-9        |              | 315                  |
| 157       | 235 | 9D  |              | 024                   | 11-4-9        |              | 316                  |
| 158       | 236 | 9E  |              | 076                   | 6-8-9         |              | 317                  |
| 159       | 237 | 9F  |              | 341                   | 11-0-1-9      |              | 320                  |
| 160       | 240 | A0  |              | 101                   | 12-0-1-9      |              | 321                  |
| 161       | 241 | A1  | ~            | 102                   | 12-0-2-9      | ~            | 176                  |
| 162       | 242 | A2  | s            | 103                   | 12-0-3-9      | s            | 163                  |
| 163       | 243 | A3  | t            | 104                   | 12-0-4-9      | t            | 164                  |
| 164       | 244 | A4  |              | 105                   | 12-0-5-9      |              | 165                  |
| 165       | 245 | A5  | u            | 106                   | 12-0-6-9      | u            | 166                  |
| 166       | 246 | A6  | v            | 107                   | 12-0-7-9      | v            | 167                  |
| 167       | 247 | A7  | w            | 110                   | 12-0-8-9      | w            | 170                  |
| 168       | 250 | A8  | x            | 111                   | 12-1-8        | x            | 171                  |
| 169       | 251 | A9  | y            | 121                   | 12-11-1-9     | y            | 172                  |
| 170       | 252 | AA  | z            | 122                   | 12-11-2-9     | z            | 322                  |
| 171       | 253 | AB  |              | 123                   | 12-11-3-9     |              | 323                  |
| 172       | 254 | AC  |              | 124                   | 12-11-4-9     |              | 324                  |
| 173       | 255 | AD  |              | 125                   | 12-11-5-9     |              | 325                  |
| 174       | 256 | AE  |              | 126                   | 12-11-6-9     |              | 326                  |
| 175       | 257 | AF  |              | 127                   | 12-11-7-9     |              | 327                  |

| CHAR CODE |     |     | ASCII        |                       |               | EBCDIC       |                      |
|-----------|-----|-----|--------------|-----------------------|---------------|--------------|----------------------|
| Dec       | Oct | Hex | Cntl/<br>Gph | to<br>EBCDIC<br>(Oct) | Hollerith     | Cntl/<br>Gph | to<br>ASCII<br>(Oct) |
| 176       | 260 | B0  |              | 130                   | 12-11-8-9     |              | 330                  |
| 177       | 261 | B1  |              | 131                   | 11-1-8        |              | 331                  |
| 178       | 262 | B2  |              | 142                   | 11-0-2-9      |              | 332                  |
| 179       | 263 | B3  |              | 143                   | 11-0-3-9      |              | 333                  |
| 180       | 264 | B4  |              | 144                   | 11-0-4-9      |              | 334                  |
| 181       | 265 | B5  |              | 145                   | 11-0-5-9      |              | 335                  |
| 182       | 266 | B6  |              | 146                   | 11-0-6-9      |              | 336                  |
| 183       | 267 | B7  |              | 147                   | 11-0-7-9      |              | 337                  |
| 184       | 270 | B8  |              | 150                   | 11-0-8-9      |              | 340                  |
| 185       | 271 | B9  |              | 151                   | 0-1-8         |              | 341                  |
| 186       | 272 | BA  |              | 160                   | 12-11-0       |              | 342                  |
| 187       | 273 | BB  |              | 161                   | 12-11-0-1-9   |              | 343                  |
| 188       | 274 | BC  |              | 162                   | 12-11-0-2-9   |              | 344                  |
| 189       | 275 | BD  |              | 163                   | 12-11-0-3-9   |              | 345                  |
| 190       | 276 | BE  |              | 164                   | 12-11-0-4-9   |              | 346                  |
| 191       | 277 | BF  |              | 165                   | 12-11-0-5-9   |              | 347                  |
| 192       | 300 | C0  |              | 166                   | 12-11-0-6-9   |              | 173                  |
| 193       | 301 | C1  |              | 167                   | 12-11-0-7-9   | A            | 101                  |
| 194       | 302 | C2  |              | 170                   | 12-11-0-8-9   | B            | 102                  |
| 195       | 303 | C3  |              | 200                   | 12-0-1-8      | C            | 103                  |
| 196       | 304 | C4  |              | 212                   | 12-0-2-8      | D            | 104                  |
| 197       | 305 | C5  |              | 213                   | 12-0-3-8      | E            | 105                  |
| 198       | 306 | C6  |              | 214                   | 12-0-4-8      | F            | 106                  |
| 199       | 307 | C7  |              | 215                   | 12-0-5-8      | G            | 107                  |
| 200       | 310 | C8  |              | 216                   | 12-0-6-8      | H            | 110                  |
| 201       | 311 | C9  |              | 217                   | 12-0-7-8      | I            | 111                  |
| 202       | 312 | CA  |              | 220                   | 12-11-1-8     |              | 350                  |
| 203       | 313 | CB  |              | 232                   | 12-11-2-8     |              | 351                  |
| 204       | 314 | CC  |              | 233                   | 12-11-3-8     | J            | 352                  |
| 205       | 315 | CD  |              | 234                   | 12-11-4-8     |              | 353                  |
| 206       | 316 | CE  |              | 235                   | 12-11-5-8     | K            | 354                  |
| 207       | 317 | CF  |              | 236                   | 12-11-6-8     |              | 355                  |
| 208       | 320 | D0  |              | 237                   | 12-11-7-8     | L            | 175                  |
| 209       | 321 | D1  |              | 240                   | 11-0-1-8      | J            | 112                  |
| 210       | 322 | D2  |              | 252                   | 11-0-2-8      | K            | 113                  |
| 211       | 323 | D3  |              | 253                   | 11-0-3-8      | L            | 114                  |
| 212       | 324 | D4  |              | 254                   | 11-0-4-8      | M            | 115                  |
| 213       | 325 | D5  |              | 255                   | 11-0-5-8      | N            | 116                  |
| 214       | 326 | D6  |              | 256                   | 11-0-6-8      | O            | 117                  |
| 215       | 327 | D7  |              | 257                   | 11-0-7-8      | P            | 120                  |
| 216       | 330 | D8  |              | 260                   | 12-11-0-1-8   | Q            | 121                  |
| 217       | 331 | D9  |              | 261                   | 12-11-0-1     | R            | 122                  |
| 218       | 332 | DA  |              | 262                   | 12-11-0-2     |              | 356                  |
| 219       | 333 | DB  |              | 263                   | 12-11-0-3     |              | 357                  |
| 220       | 334 | DC  |              | 264                   | 12-11-0-4     |              | 360                  |
| 221       | 335 | DD  |              | 265                   | 12-11-0-5     |              | 361                  |
| 222       | 336 | DE  |              | 266                   | 12-11-0-6     |              | 362                  |
| 223       | 337 | DF  |              | 267                   | 12-11-0-7     |              | 363                  |
| 224       | 340 | E0  |              | 270                   | 12-11-0-8     |              | 134                  |
| 225       | 341 | E1  |              | 271                   | 12-11-0-9     |              | 237                  |
| 226       | 342 | E2  |              | 272                   | 12-11-0-2-8   | S            | 123                  |
| 227       | 343 | E3  |              | 273                   | 12-11-0-3-8   | T            | 124                  |
| 228       | 344 | E4  |              | 274                   | 12-11-0-4-8   | U            | 125                  |
| 229       | 345 | E5  |              | 275                   | 12-11-0-5-8   | V            | 126                  |
| 230       | 346 | E6  |              | 276                   | 12-11-0-6-8   | W            | 127                  |
| 231       | 347 | E7  |              | 277                   | 12-11-0-7-8   | X            | 130                  |
| 232       | 350 | E8  |              | 312                   | 12-0-2-8-9    | Y            | 131                  |
| 233       | 351 | E9  |              | 313                   | 12-0-3-8-9    | Z            | 132                  |
| 234       | 352 | EA  |              | 314                   | 12-0-4-8-9    |              | 364                  |
| 235       | 353 | EB  |              | 315                   | 12-0-5-8-9    |              | 365                  |
| 236       | 354 | EC  |              | 316                   | 12-0-6-8-9    | H            | 366                  |
| 237       | 355 | ED  |              | 317                   | 12-0-7-8-9    |              | 367                  |
| 238       | 356 | EE  |              | 332                   | 12-11-2-8-9   |              | 370                  |
| 239       | 357 | EF  |              | 333                   | 12-11-3-8-9   |              | 371                  |
| 240       | 360 | F0  |              | 334                   | 12-11-4-8-9   | 0            | 060                  |
| 241       | 361 | F1  |              | 335                   | 12-11-5-8-9   | 1            | 061                  |
| 242       | 362 | F2  |              | 336                   | 12-11-6-8-9   | 2            | 062                  |
| 243       | 363 | F3  |              | 337                   | 12-11-7-8-9   | 3            | 063                  |
| 244       | 364 | F4  |              | 352                   | 11-0-2-8-9    | 4            | 064                  |
| 245       | 365 | F5  |              | 353                   | 11-0-3-8-9    | 5            | 065                  |
| 246       | 366 | F6  |              | 354                   | 11-0-4-8-9    | 6            | 066                  |
| 247       | 367 | F7  |              | 355                   | 11-0-5-8-9    | 7            | 067                  |
| 248       | 370 | F8  |              | 356                   | 11-0-6-8-9    | 8            | 070                  |
| 249       | 371 | F9  |              | 357                   | 11-0-7-8-9    | 9            | 071                  |
| 250       | 372 | FA  |              | 372                   | 12-11-0-2-8-9 |              | 372                  |
| 251       | 373 | FB  |              | 373                   | 12-11-0-3-8-9 |              | 373                  |
| 252       | 374 | FC  |              | 374                   | 12-11-0-4-8-9 |              | 374                  |
| 253       | 375 | FD  |              | 375                   | 12-11-0-5-8-9 |              | 375                  |
| 254       | 376 | FE  |              | 376                   | 12-11-0-6-8-9 |              | 376                  |
| 255       | 377 | FF  |              | 377                   | 12-11-0-7-8-9 | EO           | 377                  |

# NATIVE LANGUAGE COLLATING

APPENDIX

C

Native Language Support for the HP 3000 supports collating for a variety of native languages. A number of collating algorithms, from simple to very complex, have been employed in defining the collating sequences for these languages, depending on the requirements of the native users of the languages.

Native language collating sequences are accessed in SORT-MERGE/3000 by using the key type CHARACTER and the LANGUAGE command to define which native language collating sequence is to be used. In addition to actual native languages, an artificial language, "NATIVE-3000", has been defined which handles all language aspects in a traditional computer manner. Thus, for example, one collating sequence for NATIVE-3000 treats key of type CHARACTER the same as keys of type BYTE, i.e. collates them according to the value of the ASCII code for each character.

For a list of languages supported on your HP 3000, run NLUTIL.PUB.SYS. A list of language names and language ids will be displayed. The exact list will depend on the configuration chosen by your system manager. Configured languages may include, but are not limited to, those shown in the table below. The program NLUTIL will also offer to print the definition, including the collating sequence, of each language supported. See the Native Language Support Reference Manual for further information.

| Lang ID | Lang Name       | Char Set ID | Char Set Name |
|---------|-----------------|-------------|---------------|
| 0       | NATIVE-3000     | 0           | USASCII       |
| 1       | AMERICAN        | 1           | ROMAN8        |
| 2       | CANADIAN-FRENCH | 1           | ROMAN8        |
| 3       | DANISH          | 1           | ROMAN8        |
| 4       | DUTCH           | 1           | ROMAN8        |
| 5       | ENGLISH         | 1           | ROMAN8        |
| 6       | FINNISH         | 1           | ROMAN8        |
| 7       | FRENCH          | 1           | ROMAN8        |
| 8       | GERMAN          | 1           | ROMAN8        |
| 9       | ITALIAN         | 1           | ROMAN8        |
| 10      | NORWEGIAN       | 1           | ROMAN8        |
| 11      | PORTUGUESE      | 1           | ROMAN8        |
| 12      | SPANISH         | 1           | ROMAN8        |
| 13      | SWEDISH         | 1           | ROMAN8        |
| 41      | KATAKANA        | 1           | KANA8         |

Table C-1. Native Languages on the HP 3000



- Access option, input and output files 2-3
- ALTSEQ command 2-1
- ALTSEQ command, examples of 2-28
- ALTSEQ command, in batch mode 2-32
- ALTSEQ command, parameters of 2-4
- ASCII collating sequence, example of 2-34
- ASCII collating sequence, specified by DATA command 2-7
- ASCII collating sequence, table B-1
- ASCII/EBCDIC translation table B-1
- Available memory considerations, MERGE 6-2
- Calling MERGE, SPL example 6-12
- Cancelling a SORT or MERGE 2-9
- Cards, as input file 2-42
- CHARACTER key type 2-15
- Collating lower-upper case alphabetic characters 2-29
- Collating sequence, displaying 2-21
- Collating sequence, examining 2-21
- Colon, use in entering MPE commands 2-24
- Commands, definitions of 2-1
- Continuation character, long lines 2-1
- CONTROL Y 2-26
- Correcting keys, RESET command 2-20
- DATA command 2-1, 2-7
- Data segment size, running MERGE 4-2, 6-3
- Data segment size, running SORT 3-3
- Data type, input data 2-7
- Default values for parameters, RESET 2-8
- Defining special collating sequence 2-4
- Display file 2-3
- Displaying collating sequence 2-21
- Displaying translation table 2-21
- EACH, (ALTSEQ) 2-4
- EBCDIC collating sequence, example of 2-34
- EBCDIC collating sequence, specified by DATA command 2-7
- EBCDIC/ASCII translation table B-1
- END command 2-1, 2-9
- End of Data command, EOD 2-26
- Equivalencing characters, 2-4
- Error messages, display of from SPL program 3-13
- Error messages, list of A-1
- Error messages, retrieval of, MERGE 4-11
- Errorproc, SPL example 3-25, 3-27
- EXIT command 2-2, 2-10
- Failure parameter, example of 4-25
- Failure parameter, SORTINITIALF, SPL example 3-36
- File equations, use of 2-3
- File equations, using with SORT 2-42
- File, definitions of 2-3
- Files, default access options 2-3
- Files, display 2-3
- Files, input 2-3
- Files, list 2-3
- Files, output 2-3
- Files, scratch 2-3
- Files, text 2-3
- Flowchart, SORT intrinsics 3-2
- Input data type, DATA command 2-7
- Input file 2-3
- INPUT for MERGE 2-2
- INPUT for MERGE, parameters of 2-13
- INPUT for SORT 2-2
- INPUT for SORT, parameters of 2-11
- Input records, terminate using :EOD 2-27
- KEY command 1-1
- Key fields only as output files 2-44
- KEY, definition of 1-1
- Keycompare parameter, use with SORTINIT 3-24
- Keyonly parameter, example of 4-15
- LANGUAGE command 2-16A
- List file 2-3
- Listing current SORT/MERGE options, VERIFY 2-23
- Long commands, continuation character 2-1
- Memory utilization, MERGE 4-2
- Memory, considerations with MERGE 6-2
- MERGE 6-12
- MERGE operation, in batch mode 2-48
- MERGE operation, in interactive mode 2-48
- MERGE PROGRAM INTRINSICS 6-1
- MERGE program, intrinsics 4-1
- MERGE, preparation and execution when using FORTRAN/3000
- MERGE, using interactively 1-5
- MERGE, when using FORTRAN/3000 4-12
- MERGEEND 6-8
- MERGEEND, when using FORTRAN/3000 4-8
- MERGEERRORMESS 6-11
- MERGEERRORMESS, example of 4-23
- MERGEERRORMESS, when using FORTRAN/3000 4-11
- MERGEINIT 6-3
- MERGEINIT, when using FORTRAN/3000 4-3
- MERGEOUTPUT 6-7
- MERGEOUTPUT, example of 4-18A
- MERGEOUTPUT, when using FORTRAN/3000 4-7
- MERGESTAT 6-9
- MERGESTAT, example of 4-20
- MERGESTAT, when using FORTRAN/3000 4-9
- MERGETITLE 6-10
- MERGETITLE, example of 4-21
- MERGETITLE, when using FORTRAN/3000
- Modification specifications, examples 2-27
- Modification specification, types of 2-28
- Modifying collating sequence 2-4
- MPE commands, entering while in SORT/MERGE 2-24
- Multiple input files, with SORT 2-43
- Multirecord files, SPL example with MERGE 6-15
- Native Language collating 1-2, C-1
- Native Language Support, introduction C-1
- Native Language Support, stand-alone 2-15, 2-16A, 2-47A, 2-49A
- Native Language Support when using FORTRAN/3000 3-7, 3-31A, 4-6, 4-18
- Native Language Support when using SPL/3000 5-7, 5-21, 6-6, 6-17
- NOBUF files, SPL example with MERGE 6-15
- Options, listing of during SORT/MERGE 2-23
- Ordering sequence 1-2, 2-16
- Ordering sequence, to display using SHOW 2-22
- Output file 2-3
- OUTPUT for MERGE 2-19

OUTPUT for SORT 2-17  
 Overriding default file access options 2-3  
 Preprocessor parameter, example of 4-13  
 Prompt character, interactive use 2-1  
 Prompt file 2-3  
 RESET command 2-21  
 Scratch file 2-3  
 Segsize, in preparation for running MERGE 4-2, 6-3  
 Segsize, in preparation for running SORT 3-3  
 Sequence, of MERGE intrinsics 4-2  
 Sequence, of SORT intrinsics 3-2  
 SHOW command, examples of 2-22, 2-34  
 SORT operation, using multiple input files 2-44  
 SORT operation, using terminal as output file 2-42  
 SORT operation, with cards as input file 2-43  
 SORT program, intrinsics 3-1, 5-1  
 SORT program, preparation and execution when using  
 SPL/3000 5-3  
 SORT, preparation and execution when using  
 FORTRAN/3000 3-3  
 SORT, using interactively 1-2  
 SORTEND, when using FORTRAN/3000 3-10  
 SORTEND, when using SPL/3000 5-10  
 SORTERRORMESS, when using  
 FORTRAN/3000 3-13  
 SORTERRORMESS, when using SPL/3000 5-13  
 SORTINIT with altseq Parameter, example of 5-15  
 SORTINIT, SPL examples 3-22 to 3-28  
 SORTINIT, when using FORTRAN/3000 3-4  
 SORTINIT, when using SPL/3000 5-4  
 SORTINITIAL without inputfile and outputfile,  
 example of 5-19  
 SORTINITIAL, when using SPL/3000 5-14  
 SORTINITIALF, SPL example 3-36  
 SORTINITIALF, when using FORTRAN/3000 3-14  
 SORTINPUT, SPL example 3-30  
 SORTINPUT, when using FORTRAN/3000 3-8  
 SORTINPUT, when using SPL/3000 5-8  
 SORTOUTPUT, example of 5-18  
 SORTOUTPUT, SPL example 3-32  
 SORTOUTPUT, when using FORTRAN/3000 3-9  
 SORTOUTPUT, when using SPL/3000 5-9  
 SORTSTAT, SPL example 3-33  
 SORTSTAT, when using FORTRAN/3000 3-11  
 SORTSTAT, when using SPL/3000 5-11  
 SORTTITLE, SPL example 3-34  
 SORTTITLE, when using FORTRAN/3000 3-12  
 SORTTITLE, when using SPL/3000 5-12  
 Special collating sequences, examples 2-27  
 Statistics parameter, example of 4-17  
 Statistics parameter, SPL SORT example 3-29  
 Statistics, display from SPL program 3-29  
 Status, obtain using CONTROL Y 2-26  
 Summarizing current options, VERIFY 2-23  
 TABLE parameter, used with SHOW 2-23  
 Terminal, use as input file, example 2-41  
 Terminal, use as output file, example 2-41  
 Terminating the SORT/MERGE program 2-8  
 Termination of input records 2-26  
 Text file 2-3  
 Translation table, displaying 2-21  
 Translation table, editing of 3-17  
 Translation table, initializing 2-7  
 Translation, ASCII/EBCDIC B-1  
 User data segment size, running MERGE 4-2, 6-3  
 User data segment size, running SORT 3-3  
 User defined collating sequences 2-4  
 User defined collating sequences, examples 2-27  
 VERIFY command 2-24