DECSYSTEM

SORT
User's Guide

Order No. DEC-20-USTGA-A-D

20

digital

SORT
User's Guide

Order No. DEC-20-USTGA-A-D

digital equipment corporation · maynard, massachusetts

First Printing, January 1976

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

CONTENTS (Cont.)

TABLES

PREFACE

This document is written for those who have a need to sort data into a specific sequence. The reader need not have programming skills but should be familiar with computer operations and at least have a knowledge of data types and recording modes. Important terms are defined in the Glossary.

This document reflects the software as of version 2.

Within this manual, references are made to the following documents:

COBOL REFERENCE MANUAL – DEC-20-LCRMA-A-D

MONITOR CALLS MANUAL – DEC-20-OMRMA-A-D

HARDWARE REFERENCE MANUAL – EK-DEC10-RF-001

CHAPTER 1
THE SORT PROGRAM

1.1 INTRODUCTION

SORT is a stand-alone utility program that arranges the records of one or more files according to a user-specified sequence. The user designates the keys on which the records are sorted from one or more fields within a record. The keys can be in either ascending or descending order. SORT compares the values of all records in the key fields. Then it arranges the records in the specified sequence and merges them into a single output file.

Two procedures are performed by the SORT program, sorting and merging. During the sorting process, the input files are read, and the records are sifted into runs. These runs are output to one or more temporary files. Up to 15 temporary files can be used for scratch devices. If the number of records is small, the run may remain in core. The merging process combines the runs of the temporary files according to the user-specified keys. There may be intermediate merges depending on the number of runs in the temporary files, but there is one final output merge.

SORT runs on disk units and always provides a stable sort in both timesharing and batch modes. In a stable sort, two or more records with exactly the same key will stay in the same relative order after being sorted.

All ASCII, COBOL, and FORTRAN files can be sorted.

1.2 HARDWARE AND SOFTWARE ENVIRONMENTS

The normal minimum hardware required for small sorts is on disk unit and 128 pages of memory. The more disk units and memory available, the faster the sort. Although tape units can be used for the input and output files, disks must be used for the temporary files. If the source files are on disk, the user must have enough temporary disk space to hold the input files plus two copies of these files. If the source files are on magnetic tape, at least one tape drive is required.

The minimum memory required is the sum of the following:

1. Double buffers for
   a. The input devices,
   b. The output device,
   c. One temporary file on disk.
2. At least 16 but preferably 128 records, and
3. One record from each temporary file.

The software required to run SORT is MACRO version 50 and LINK version 2 or later.

SORT uses the SCAN program to read and interpret the user's command string.

1.3 DATA TYPES, RECORDS, AND RECORDING MODES

In order to be aware of the position and size of key fields, the user must know the data types, records, and recording modes of his files.

1.3.1 Data Types

SORT accepts five basic types of data in key fields:

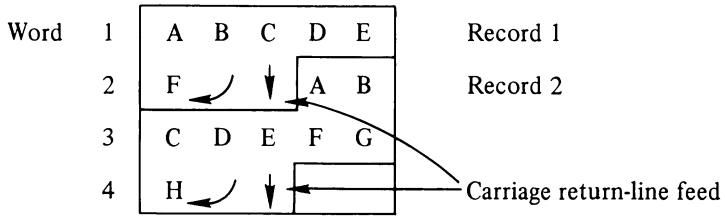
1. Alphanumeric	A combination of characters consisting of the alphabet, numbers, and symbols.
2. Numeric	One or more numbers.
3. COMPUTATIONAL or COMP	Fixed-point binary.
4. COMPUTATIONAL1 or COMP1	Floating-point binary.
5. COMPUTATIONAL3 or COMP3	Packed decimal.

All five data types are represented by switches in the SORT program and can be specified in the command string. Note that alphanumeric comparisons are performed according to either the ASCII or EBCDIC Character Collating Sequence. This sequence is listed in Appendix C. Numeric, COMPUTATIONAL, COMPUTATIONAL1, and COMPUTATIONAL3 compares are determined by comparing the results of one algebraic value to another.

1.3.2 Records

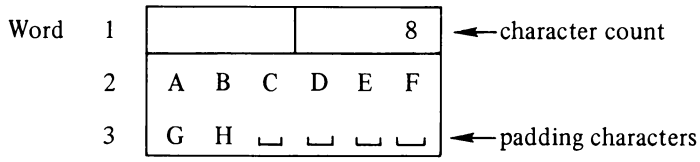
SORT is capable of reading and writing four types of records: ASCII, SIXBIT, EBCDIC, and binary. An ASCII file is made up of characters packed five per 36-bit words, left-justified. The last bit of each word is ignored on input and is zero on output unless it is a sequence number. An ASCII record does not have to begin or end on a word boundary. The user can terminate the record with a carriage return. (A line feed usually follows the carriage return.) Any vertical format separator (i.e., carriage return/line feed, form feed, vertical tab) is ignored on input.

The following diagram illustrates the format of an ASCII record in an unaligned file:



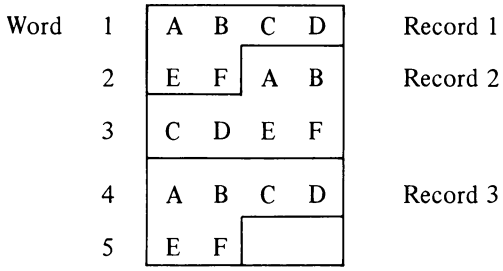
In a SIXBIT file, characters are packed six per 36-bit word. SIXBIT is a compressed form of ASCII, but, unlike ASCII, a record must start on a word boundary. A SIXBIT record contains a set of contiguous words; the right half of the first word has the number of characters in the record. To ensure that the record ends on a word boundary, the last word is padded with blanks, if necessary. When determining the size of the record, the user must take into consideration the actual number of characters in the record, one additional word to contain the character count, and any additional characters necessary for padding to reach a word boundary.

The following diagram illustrates the format of a SIXBIT record:



An EBCDIC file is made up of characters packed four per 36-bit word, left justified; it may have record and block headers. An EBCDIC record does not have to begin or end on word boundaries. For more information on EBCDIC records, see Chapters 4 and 5 of the *COBOL REFERENCE MANUAL.*

The following diagram illustrates the format of an EBCDIC record in a file:



A BINARY record is made up of 36-bit words, starting at a word boundary, and always fixed length.

1.3.3 Recording Modes

The recording mode describes how the data files are read or written within a program. Only one recording mode of the set ASCII, SIXBIT or EBCDIC can be specified for all input files in the command string; i.e., /ASCII or /SIXBIT can be specified, but not both. /BINARY can be specified with any of the other three, in which case the actual I/O is binary; internally, however, the file is as described by the other switch.

ASCII	five 7-bit characters to one 36-bit word.
SIXBIT	six 6-bit characters to one 36-bit word.
EBCDIC	four 9-bit bytes to one 36-bit word.
BINARY	36 bits to one word.

If the user specifies two or more recording modes in the same command string (with the exception of BINARY), he will receive the following message:

```
?SRTMSC MODE SWITCH CONFLICT
```

The user must run SORT again using only one recording mode.

1.4 BLOCKED AND UNBLOCKED FILES

Blocking files is the method of grouping records into logical blocks with a specified number of records in each block. A blocking factor is the limit of the data records which can be contained in a given block on tape. SORT accepts both blocked and unblocked files in the same command string. The /BLOCK switch, described in Chapter 3, designates the blocking factor for each file. For more information concerning blocking files, refer to the *COBOL REFERENCE MANUAL*.

CHAPTER 2
HOW TO USE SORT

2.1 INFORMATION NEEDED TO RUN SORT

The user must know the following about the input files in order to sort and merge them:

1. The recording mode,
2. The record size,
3. The blocking factor, and
4. The label status.

The next step is to determine how the files are to be sorted. Within each record, the user must know:

1. The location of the key field,
2. The data type of each key field, and
3. The order of each key field (ascending or descending).

Finally, for the output file, the user must decide:

1. The blocking factor,
2. The word-alignment (if recording mode is ASCII),
3. The label status, and
4. The record size.

In addition, the user can estimate the amount of memory needed and specify that amount when running SORT. Otherwise, SORT will use its default algorithm to determine the amount of memory required. This method may not be the most efficient. See section 2.1.3 for more information on specifying memory.

2.1.1 General Command Format

The general command format is

output/switch(es)=input/switch(es),input/switch(es),input/switch(es),...

where

output	=	the file specification for the output file.
input	=	the file specification for each input file.
switch(es)	=	one or more switches for the input files, output file, sorting process, and sometimes SCAN. (Refer to Chapter 3 for a complete list of SORT switches and Appendix B for SCAN switches.)

A file specification takes the form:

dev:name.typ[proj,prog]

If dev: is omitted, the system uses DSK:. If [proj,prog] is omitted, the system uses the user's logged-in directory. There is no default for the file type. See Appendix E, "Accessing Another User's File".

The command string can be continued on more than one line by inserting a hyphen at the end of the line to be continued. When a hyphen is used, SORT outputs a number sign (#) on the next line to verify that the continuation has been noted.

```
*output=input/switch(es),input/switch(es),input/switch(es)-
#
```

The user must specify the following in the command string:

1. Only one output file,
2. The equal sign (=),
3. At least one input file,
4. At least one key field, and
5. At least one record size.

One or more input files can be specified (separated by commas) but only one output file can be specified.

The user specifies the keys in the command string with the switches described in Chapter 3. Any number of keys can be specified for sorting; they are used in the order in which they appear in the command string, from left to right. Because the individual files are sorted and merged into one output file, the user must be sure that the keys of each record of each input file have the same characteristics (such as location, length, and data type). Only one recording mode can be given for all the input files; therefore, an ASCII file cannot be combined with a SIXBIT file.

Another method of supplying commands to SORT is to build a command file consisting of several SORT command strings. Then when SORT calls for a command string, the user replies with

@file specification

SORT will read all the commands from the command file and will execute all the sorts (if they contain no errors).

2.1.2 Devices

Unless the user specifies another device, the system reads and writes disk files.

If the user has multi-reel files with standard labels, he can specify each device as a separate file in the command string. SORT will treat each reel as a separate file and continue the file when it finds the next sequential label.

During the sorting process, the input files are allocated to temporary scratch devices. The SORT program uses as many of these temporary devices as needed up to a maximum of 15.

The default temporary scratch device is DSK. SORT tries to write each temporary file on a separate unit.

2.1.3 Specifying Memory

There are three methods of memory allocation in the SORT program:

1. @SORT nP
2. /CORE switch (see Chapter 3), and
3. SORT default core algorithm.

If the user does not specify the amount of memory with methods 1 or 2, SORT uses the default memory algorithm to expand to the necessary amount of memory. In this instance, the following informational message is printed after the user types the command string:

[SRTXPN EXPANDING TO nP]

At this point, if SORT has allocated too much memory, the user can make the sort more efficient by specifying less with methods 1 or 2.

If, however, the user does not specify enough memory, he receives this message:

%SRTNCS NOT ENOUGH CORE SPECIFIED

SORT will then use its default memory algorithm to expand memory.

2.1.4 Terminal Output from SORT

After the sort/merge phases are complete and no errors are encountered, SORT provides the following information:

```
SORTED n RECORDS
n KEY COMPARISONS,          n PER RECORD
n RUNS, BIAS n.nn
hh:mm:ss CPU TIME,          n MS PER RECORD
hh:mm:ss ELAPSED
```

where

n	is a number, and
hh:mm:ss	is a set of numbers representing time in the form of hours, minutes, and seconds.

The number of records sorted tells the user how many records have been read from the input files and output to the output file. The user is told how many times the keys were compared and how many comparisons were made per record.

A run is made up of the records that SORT has output in sequence to a work file. The number of runs determines how many temporary files were used.

If several runs occurred during the sort, the bias is also printed. The bias is the ratio of an average run to the span (span being the total number of records that will fit in memory at any given time during a sort). Thus, the bias gives the user a measure of how ordered the original files were: for a truly random file, the bias should be approximately 2.0.

The CPU and ELAPSED times begin at the start of the sort and correspond to the values returned by the INFORMATION (ABOUT) PROGRAM-STATUS command. (Refer to the *DECsystem-20 USER'S GUIDE* for an explanation of this command.)

CHAPTER 3
SWITCHES

3.1 SWITCHES AVAILABLE IN SORT

The command string can contain numerous switches, some required, some optional. Some switches apply to the input files, the output file, or both input and output files. Other switches apply to the sorting process. The user specifies the keys to be extracted and compared by inputting the various switches described in this chapter. The limitations are given with the switches and in Appendix A. (Additional switches can be specified in the command string to SCAN. They are listed in Appendix B.)

Each switch must be preceded by a slash (/); arguments are separated from switches with colons; for example,

```
/BLOCKED:n
```

Switches can be abbreviated. They need consist of only those letters that are required to make the switch unique. (Users are encouraged to use at least 3 letters to prevent conflict with switches in future implementations.)

3.1.1 The Device Handling Switches

The /REWIND switch rewinds the specified device *before* the specified file is read or written.

the /UNLOAD switch unloads the specified device *after* the specified file is written.

The format of these switches is

```
dev1:output file spec/UNLOAD = dev2:input file spec/REWIND
```

Each of these switches applies only to the file specification that it follows.

The /INDUSTRY switch reads or writes a magnetic tape in industry compatible mode. It is ignored on all other devices.

The format of this switch is

```
/INDUSTRY
```

For information about industry compatible mode, see the *Monitor Calls Manual* and the *Hardware Reference Manual.*

3.1.2 The /CORE Switch

The /CORE switch applies to the sorting process. Its purpose is to specify the amount of memory that the low segment of SORT will use during execution.

This switch has the following format:

/CORE:nP	The amount of memory to be used by SORT's low segment in pages (512 words).

The /CORE switch can only be specified once, and it can be placed anywhere in the command string. When the user allocates memory with this switch or with @ SORT nP, SORT tries to use this amount. However, if the memory is not large enough, SORT tries to expand it. If this fails, SORT terminates the job. The user must restart SORT and allocate more memory.

If the user does not allocate memory in any way, SORT uses its default memory algorithm. The amount allocated depends on system parameters.

3.1.3 The /KEY Switch

The /KEY switch specifies a key field. Any number of keys can be specified in the command string. The keys are utilized in the order in which they are given, left to right.

The format of this switch is

/KEY:F:L:O

where: F is a decimal integer giving the number position of the first character in the key. Alphanumeric and numeric characters are counted starting at position 1. COMP and COMP1 key positions are given by specifying the leftmost character position. For SIXBIT, this is always a value of 6n + 1 (i.e., 1, 7, 13, 19). For EBCDIC, it is a value of 4n + 1 (i.e., 1, 5, 9, 13, 17). For ASCII, it is a value of 5n + 1 (i.e., 1, 6, 11, 16). These last two specifications – for ASCII and EBCDIC – are not recommended for binary files, since they may cause a loss of data. (COMP and COMP1 keys must begin on a word boundary.)

L is a decimal integer giving the length of the key in characters or digits. COMP and COMP1 keys are specified in digits; SORT interprets 10 digits or less as 1 word and 11 digits or more as 2 words. COMP3 is specified in digits, not including the sign.

O is the letter A or the letter D, representing the words "ascending" or "descending". This argument determines the order of the sort on this key only. The default sequence is ascending order.

NOTE

If a user has ASCII files with sequence numbers, he must take them into account when determining the starting position of the key. The sequence numbers are considered part of the file (six characters).

The key field cannot be larger than the size of the record. That is, the sum of the starting position and length cannot exceed the size of the record.

The /KEY switch applies to the sorting process, and all keys specified apply to all input files. This switch may be given any place in the command string. The user may include the data type and sign switches immediately following the key in the string. If the data type and sign switches are specified, they apply only to the key designated by the preceding /KEY switch.

This is no default for the first two fields of this switch; the starting position and length are required.

3.1.4 The Data Type Switches

There are five data type switches available in SORT. These switches define the data type of the key field to be compared.

The format of these switches is

/ALPHANUMERIC	The key is alphanumeric.
/NUMERIC	The key is numeric display.
/COMP	The key is COMPUTATIONAL – fixed-point binary.
/COMP1	The key is COMPUTATIONAL1 – floating-point binary.
/COMP3	The key is COMPUTATIONAL3 – packed decimal.

NOTE

SORT compares COMP and COMP1 keys identically; therefore, the user can specify either one and receive the same results.

Each key may have its own data type, but only one type can be specified per key. If the user includes a data type switch in the command string, he must place it after the /KEY switch to which it applies. If a data type is not specified, the default type is normally /ALPHANUMERIC. However, if one of the sign switches (described next) is given, the default data type is /NUMERIC.

3.1.5 The Sign Switches

NUMERIC, COMP, COMP1, and COMP3 keys have an operational sign. The sign switches tell SORT whether the sign is to be used in the key comparisons.

The format of these switches is

/SIGNED	The operational sign of the key is to be used.
/UNSIGNED	The operational sign of the key is to be ignored.

Only one sign switch can be specified for each key. If the data type is given, the user must input the sign switch directly after it in the command string, i.e.,

```
output=input/KEY:F:L/NUMERIC/UNSIGNED
```

The default sign switch depends on the data type. If the user specifies /NUMERIC, /COMP, /COMP1, or /COMP3, the default is /SIGNED. If the data type is /ALPHA, the switch is not required.

3.1.6 The Recording Mode Switches

SORT can read and write four modes: ASCII, SIXBIT, EBCDIC, and binary. The user can specify a file's recording mode with one of the following switches:

/ASCII	The files are recorded in ASCII mode.
/SIXBIT	The files are recorded in SIXBIT mode.
/EBCDIC	The files are recorded in EBCDIC mode.
/BINARY	The files, if produced by COBOL, are recorded in binary mode.

The /ASCII, /SIXBIT, /EBCDIC, or /BINARY switch can be placed anywhere in the command string. Only one type of recording mode, however, can be specified; SORT does not convert from one mode to another. If more than one mode, other than BINARY, is given in the command string, SORT will not run, and the user will receive the following message:

```
?SRTMSC MODE SWITCH CONFLICT
```

At this point, the user must run SORT again.

If the key's data type is alphanumeric or numeric display, the default mode is ASCII. If the key's data type is COMP or COMP1, the default is SIXBIT. If it is COMP3, the default is EBCDIC; /EBCDIC, in fact, is the only switch that can be used with the COMP3 key. (These switches apply to both input and output files.)

CAUTION

If the /COMP or /COMP1 switch is used with the /ASCII or /EBCDIC switch, data may be lost from the file when it is recorded.

When the recording mode is ASCII, the /ALIGN switch may also be specified. See section 3.1.10.

3.1.7 The /FORTRAN Switch

The /FORTRAN switch reads and writes FORTRAN files. Its format is

```
/FORTRAN
```

If /FORTRAN is used with the /ASCII switch, the file is word aligned on output. If it is used with the /BINARY switch, the file is assumed to have LSCWs (Logical Sector Control Words).

Besides /ASCII or /BINARY, one of the following two switches may be used with the /FORTRAN switch:

```
/RANDOM
/SEQUENTIAL
```

If the /RANDOM switch is used, only the start and end LSCWs are assumed to be present, and the file is assumed to be fixed length.

If the /SEQUENTIAL switch is used, then all three LSCWs are possible, and the file may be variable length. FORTRAN binary files are sequential by default.

Unless the /FORTRAN switch is used, a file specified with /BINARY is assumed to be COBOL binary (i.e., fixed length without header words).

FORTRAN image binary files can be sorted as if they were produced by COBOL.

3.1.8 The /ALIGN Switch

If the user's files are recorded in ASCII mode, he can specify that each record start at a word boundary in the output file. This will make the output file word-aligned. By including the /ALIGN switch in the command string, the user increases the rate of data transfer, but produces a larger than normal output file.

The /ALIGN switch is not needed with files that have sequence numbers. Each sequence number is the beginning of a new word. (These files can be variable length.)

This switch has the following format:

```
/ALIGN
```

The output file will not be word-aligned if this switch is omitted (unless it has sequence numbers). Therefore, each record will start immediately after the previous one. The /ALIGN switch is useful if the output file is to be printed.

This switch applies only to the output file, and it can be placed anywhere in the command string.

3.1.9 The Record Size and Record Length Switches

The record size must be specified at least once in the command string. The size is designated by the / /RECORD switch.

The format of this switch is as follows:

```
/RECORD:n
```

where: n is a decimal integer designating the number of characters in the record.

To determine the record size, the user should not include the character count word in SIXBIT records or the carriage return/line feed characters in ASCII records. When the user runs SORT, if the record length is larger than the size specified by n, the record will be truncated. If the record length is variable, the user should use the length of the largest record as n, or data will be lost.

The /RECORD switch applies to either input or output. If it is placed on the output side, it applies to the entire command string. Placing /RECORD:n before a filename on the input side also determines the default for the entire string. If the user specifies this switch after a filename, it applies only to that particular file. For example, in the following

```
OUTPUT=/RECORD:120 AFILE,BFILE/RECORD:150,CFILE
```

AFILE, CFILE, and the output file have a record length of 120, while BFILE has a record length of 150.

If the user does not specify the record length, SORT will not execute the sort/merge, and the user will receive the following message:

```
?SRTRSR RECORD SIZE REQUIRED
```

A record has either a variable length or a fixed length. The following switches identify the record length to SORT:

/VARIABLE	The file has variable length records.
/FIXED	The file has fixed length records.

These switches apply to either input or output. The switch placed on the output side applies to the entire command string. Placing the switch before a filename on the input side also determines the default for the entire string. If the user specifies this switch after a filename, the length applies only to the particular file preceding it.

For example, if the user inputs

```
OUTPUT=/RECORD:n/FIXED AFILE,BFILE,CFILE
```

the record length for the entire command string is fixed.

If the recording mode is SIXBIT or EBCDIC, the record length is assumed to be fixed; if the recording mode is ASCII, the record length is assumed to be variable.

3.1.10 The /BLOCKED Switch

SORT accepts both blocked and unblocked files for sorting and merging. The user has the option of defining his files with the /BLOCKED switch. (For more information on blocking, refer to the *COBOL REFERENCE MANUAL.)*

This switch has the following format:

```
/BLOCKED:n
```

where: n is a decimal integer determining the blocking factor for the file. (The blocking factor is the number of logical records per logical block.)

The /BLOCKED switch applies to both input and output. If it is omitted from the command string, all the input files and the output file are unblocked. If any /BLOCKED switches are specified, and the user wishes to indicate that one file is unblocked, he inputs /BLOCKED:0 after that particular filename.

/BLOCKED:n placed before a filename determines the blocking factor for the entire string, input and output. If the user specifies the blocking factor on the output side only, the input files will have the same blocking factor as the output file. If the user places the switch after a filename, the blocking factor will refer only to that particular file. When the user does not specify a blocking factor for the output file, SORT uses the first /BLOCKED switch in the string as the blocking factor for the output file. For example,

```
OUTPUT=/BLOCKED:25 AFILE,BFILE,CFILE,DFILE/BLOCKED:10
```

AFILE, BFILE, CFILE, and the OUTPUT file have the same blocking factor of 25, while DFILE has a blocking factor of 10.

3.1.11 The /LABEL Switch

SORT provides a switch to designate label status of the files, the /LABEL switch.

This switch has the following format:

/LABEL:STANDARD	The file has COBOL-standard labels. SORT will read and write them on magnetic tape.
/LABEL:NONSTANDARD	The file has non-standard labels. SORT will by-pass them on input and omit them on output. These labels are assumed to be the size of one physical record.
/LABEL:OMITTED	The file has no labels. SORT will not look for them on input or write them on output.

The /LABEL switch applies either to input or output. If the user specifies this switch on the output side, it determines the default for the input side also. A /LABEL switch placed before a filename on the input side also determines the default for the entire string. If it is placed after an input filename, it will apply only to that particular file.

NOTE

EBCDIC files can be read only with omitted or non-standard labels, and written only with omitted labels.

NOTE

Labels on disk are ignored because these devices are directory devices.

If the user omits the /LABEL switch, SORT assumes the following:

1. Each file on magnetic tape has standard labels.
2. Each file on any other device has no labels.

If the user inputs the following string

```
OUTPUT=AFILE/LABEL:OMITTED,/LABEL:STANDARD BFILE,CFILE,DFILE
```

BFILE, CFILE, and DFILE have standard labels, while AFILE has no labels. The output file has no labels because the first /LABEL switch on the input side is /LABEL:OMITTED.

CHAPTER 4
EXAMPLES

This chapter is a compilation of various SORT examples using the switches discussed in Chapter 3. The examples show sorts from 10 records up to 1,956 records.

Each example describes the following characteristics in the command string:

1. Input File(s)
2. Output File
3 Record Size
4. Data Type
5. Recording Mode
6. Description of Key(s)

where

F = The starting position of the key,
L = The length of the key, and
O = The order of the sort, ascending or descending.

The number sign (#) indicates that the continuation has been noted.

4.1 SORTING 10 RECORDS (Examples 1 and 2)

```
*XS0010.OUF=/RECORD:36/FIXED/SIXBIT/KEY:7:6-
#/COMP/KEY:13:3/ALPHA XS0010.INF

[SRTXPN Expanding to 37P]
Sorted 10 Records
18 KEY comparisons,        1.80 per record
0 Runs
 0:00:00 CPU time,        73.90 MS per record
 0:01:40 Elapsed
*
```

The command string in Example 1 designates the following:

1. Input File(s) – XS0010.INF
2. Output File – XS0010.OUF
3. Record Size – 36 and Fixed
4. Data Type – Key 1 is COMP; Key 2 is ALPHA

5. Recording Mode — SIXBIT
6. Description of Key(s)

   KEY:7:6 — F is the first character of the second word.
   L is 6 characters.
   O is ascending (default).

   KEY:13:3 — F is the first character of the third word.
   L is 3 characters.
   O is ascending (default).

After the user has run SORT and the asterisk (*) appears, he types in the command string. Because the string is long, he continues it with a hyphen and carriage return. SCAN responds with a number sign (#), and the user continues the command string.

Since the user does not allocate memory in the string, SORT uses the default core algorithm to expand memory to 137P.

Another example is:

```
*XS0012.OUF=/RECORD:36/FIXED/SIXBIT/KEY:16:3/NUMERIC-
#/KEY:19:6/COMP1 XS0012.INF

[SRTXPN Expanding to 37P]
Sorted 10 Records
23 KEY comparisons,      2.30 per record
0 Runs
 0:00:00 CPU time,       63.60 MS per record
 0:01:43 Elapsed
*
```

The command string in Example 2 designates the following:

1. Input File(s) — XS0012.INF
2. Output File — XS0012.OUF
3. Record Size — 36 and Fixed
4. Data Type — Key 1 is Numeric; Key 2 is COMP1
5. Recording Mode — SIXBIT
6. Description of Key(s)

   KEY:16:3 — F is the fourth character of the third word.
   L is 3 characters.
   O is ascending (default).

   KEY:19:6 — F is the first character of the fourth word.
   L is 6 characters
   O is ascending (default).

4.2 SORTING 40 RECORDS (Example 3)

```
*FILE3.DAT/R:6/VA/ASC=FILE2.DAT/ALPHA,FILE1.DAT/NUM/K:2:2:D/K:2:2:D

[SRTXFN Expanding to 191F]
Sorted 40 Records
217 KEY comparisons,      5.43 per record
0 Runs
 0:00:04 CPU time,        120.75 MS per record
 0:02:12 Elapsed
*
```

The command string in Example 3 contains the following:

1. Input File(s) – FILE2.DAT and FILE1.DAT
2. Output File – FILE3.DAT
3. Record Size – 6 and Variable
4. Data Type – Key 1 is Alpha; Key 2 is Numeric
5. Recording Mode – ASCII
6. Description of Key(s)

   KEY:2:2:D – F is the second character of the first word.
   L is 2 characters.
   O is descending.

   KEY:2:2:D – Same as first Key.

This string sorted 20 records from two files into one. The data type ALPHA need not have been specified by the user since it is the default.

4.3 SORTING 641 RECORDS (Example 4)

```
*106.OUF/R:30/VA/ASC/K:25:2:D/NUM/UNSIGNED/K:13:3/ALPHA-
#/K:16:3:D/NUM=106.INF

[SRTXPN Expanding to 115P]
Sorted 641 Records
5668 KEY comparisons,    8.84 per record
0 Runs
 0:00:03 CPU time,       5.27 MS per record
 0:04:29 Elapsed
*
```

The command string in Example 6 specifies the following:

1. Input File(s) – 106.INF
2. Output File – 106.OUF
3. Record Size – 30 and Variable
4. Data Type – Key 1 is NUM; Key 2 is Alpha; Key 3 is NUM.
5. Recording Mode – ASCII
6. Description of Key(s)

KEY:25:2:D	– F is the first character of the fifth word. L is 2 characters. O is descending.
KEY:13:3	– F is the first character of the third word. L is 3 characters. O is ascending (default).
KEY:16:3:D	– F is the fourth character of the third word. L is 3 characters. O is descending.

4.4 SPECIFYING A COMMAND FILE (Examples 5 and 6)

Another method of inputting a command string is to specify a command file to SORT. This method is discussed in Chapter 2. The following example calls command file "A" to be sorted.

```
*@A

[SRTXPN Expanding to 191P]
Sorted 655 Records
5288 KEY comparisons,    8.07 per record
0 Runs
 0:00:03 CPU time,       4.81 MS per record
 0:00:34 Elapsed
*
```

This is the command string contained in command file "A".

```
@TY A.CCL
OU3.FIL=IN3.FIL/REC:510/K:1:9/ASC/ALPHAN
```

After SORT prompted with an asterisk (*), the user typed an at sign (@) and the file specification. The following information was designated in the command string:

1. Input File(s) – IN3.FIL
2. Output File – OU3.FIL
3. Record Size – 510 and Variable
4. Data Type – Alphanumeric
5. Recording Mode – ASCII
6. Description of Key(s)

   KEY:1:9 – F is the first character in the first word.
   L is 9 characters.
   O is ascending (default).

Note that the record length is variable because that is the default for an ASCII recording mode.

In Example 6, the same name for the command file was used: "A"; however, the file was altered to contain 1,960 records instead of 655.

```
*@A

[SRTXPN Expanding to 191P]
Sorted 1960 Records
19784 KEY comparisons,   10.09 per record
2 Runs, Bias 1.32
 0:00:11 CPU time,       5.70 MS per record
 0:02:04 Elapsed
*
```

This is the command string contained in the command file. It is the same as in Example 5.

```
@TY A.CCL
OU3.FIL=IN3.FIL/REC:510/K:1:9/ASC/ALPHAN
```

The string included the following information:

1. Input File(s) – IN3:FIL
2. Output File – OU3.FIL
3. Record Size – 510 and Variable
4. Data Type – Alphanumeric
5. Recording Mode – ASCII
6. Description of Key(s)

   KEY:1:9 – F is the first character of the first word.
   L is 9 characters.
   O is ascending (default).

The string is exactly the same as Example 5; however, the number of records is about three times as large. Notice the difference in the amount of Key comparisons.

Note that unlike the previous examples, this example has 2 runs instead of 0. This means that the sort was large enough to require temporary files and did not remain in memory.

4.5 SUMMARY OF EXAMPLES

Examples 1 through 6 have many things in common. In each case:

1. The default memory algorithm expands memory.
2. The disk priorities are normal.
3. All files are unblocked.
4. SORT takes care of allocating temporary files when needed.
5. All ASCII output files contain characters following one right after the other (not word-aligned), unless the input files have sequence numbers.
6. All input and output files are to and from disk.
7. Labels are not needed because magnetic tapes were not used.

CHAPTER 5
MESSAGES

5.1 SORT MESSAGES

SORT conforms to the Digital standard for system diagnostic messages and error codes. Table 5-1 gives the conventions used in these messages.

Table 5-1
Message Conventions

Convention	Meaning
dev	A legal device name.
file structure name	A legal file structure name.
file.ext	A legal filename and extension.
addr	A user address.
n	A number.
abc	A disk unit or drive.
x	An alphabetic character.
switch	A switch.

The formats of the messages are:

```
?SRTXXX    text
%SRTXXX    text
[SRTXXX    text]
```

where

?	=	a fatal error – the program must be run again.
%	=	a warning message – the program will continue.
[ ]	=	an informational message – the program will continue.
SRT	=	The SORT mnemonic.
XXX	=	The three letter mnemonic for the message.
text	=	The explanation of the message.

The following is a list of messages that can be issued from SORT. They are in alphabetical order (ascending) according to the six character code.

?SRTARL ASCII RECORD LENGTH INCORRECT

One of the following occurred:

1. The user supplied an incorrect record length (i.e., included carriage return/line feed in the count or did not count sequence numbers).
2. The user specified a fixed length when the record was variable.
3. The record is not ASCII.

?SRTBNV BINARY MODE DOES NOT SUPPORT VARIABLE LENGTH RECORDS

The /BINARY switch, used with the /ASCII, /SIXBIT, or /EBCDIC switch, does not allow variable length records.

?SRTCWB COMPUTATIONAL KEY MUST BE ON A WORD BOUNDARY

In the /KEY switch, the user specified the starting position of a COMPUTATIONAL item as not starting on a word boundary.

?SRTDND DEVICE dev: NOT DISK. ALL SCRATCH DEVICES MUST BE DISK

The user supplied dev:/TEMP where dev is not a disk.

?SRTDNE DEVICE dev: DOES NOT EXIST

The user either supplied a device that does not exist, or misspelled a device name.

?SRTELN EBCDIC TAPE LABELS NOT SUPPORTED

SORT does not currently support EBCDIC tape labels.

?SRTFCI FORTRAN BINARY CONTROL WORD INCORRECT

An LSCW (Logical Sector Control Word) was expected but not found. The file has been corrupted.

?SRTILC ILLEGAL CHARACTER x IN NUMERIC FIELD

A character (x) other than a valid COBOL numeric character was found.

?SRTINS INPUT FILE NOT SPECIFIED

At least one input file must be specified.

?SRTIRE INPUT READ ERROR, STATUS code

An input read error occurred. The status code given defines the error. See Table 5-2, "Error Codes".

?SRTJAL JUNK IN ASCII LINE

In a variable length ASCII file, a carriage-return was not followed by a line-feed, form-feed, or vertical tab.

?SRTKAI KEY ARGUMENT ILLEGAL

The user specified something other than ASCENDING or DESCENDING with the /KEY switch.

?SRTKLR KEY LENGTH REQUIRED

The /KEY switch must be specified with the length and the starting position.

?SRTKOR KEY OUTSIDE OF RECORD

The sum of the starting position and the length of the key cannot exceed the size of the record plus one.

?SRTLNC LABEL NOT CORRECT FOR name.typ

The file name and type given in the command string do not match the label on the specified magnetic tape, or the header label is incorrect (i.e., HDR1 is missing). Either specify the correct filename or correct the label.

?SRTLRE ENTER ERROR (n). . . name.typ
LOOKUP
RENAME

An ENTER, LOOKUP, or RENAME error occurred on the specified file. The code of the error is specified by n. See Table 5-2, "Error Codes".

?SRTMGF MONITOR GETTAB FAILED

The GETTAB given by n (in octal) failed.

?SRTMSC MODE SWITCH CONFLICT

Two or more recording mode switches were given in the command string. Only one recording mode can be specified.

?SRTMUF MONITOR UUO FAILED nnn. . .

A monitor call failed. This should not happen. If it does, contact your local Software Specialist.

?SRTMUM MULTIPLE OUTPUT SPECS ONLY ON MAGTAPES

The user specified multiple output devices, one or more of which was not a magtape.

?SRTNCS NOT ENOUGH CORE SPECIFIED

The amount of memory specified with the /CORE switch or @ SORT nP was not sufficient. SORT will use its default memory algorithm to obtain the necessary amount of memory.

%SRTNLO NON-STANDARD LABEL OMITTED

The user specified /LABEL:NONSTANDARD on the output file. SORT ignores the switch and omits the labels.

?SRTNRL NAME REQUIRED WITH LABELLED MAGTAPE

The user must specify a file name and type with a labelled magtape.

?SRTNSD CANNOT SET DENSITY TO n ON dev:

The user gave a /DENSITY:n switch which cannot be satisfied on the given device (dev:).

.SRTOFF OPEN FAILED FOR DEVICE

The monitor call OPEN failed. See Table 5-2, "Error Codes".

?SRTOKR AT LEAST ONE KEY IS REQUIRED

The user must include at least one key on the command string.

?SRTONS OUTPUT FILE NOT SPECIFIED

One output file must be specified.

?SRTRIE RECORD INCOMPLETE AT E-O-F

The end of file was encountered before the record was completed.

%SRTRN1 RECORD NUMBER INCONSISTENT, n READ, m WRITTEN

This is an internal SORT error in which records have been lost.

?SRTROS REEL OUT OF SEQUENCE FOR name.typ

Either the specified file was not the first reel of a multi-reel file, or it was not one greater than the previous file of a multi-reel file.

?SRTRSR RECORD SIZE REQUIRED

The /RECORD switch must be specified at least once in the command string.

%SRTRT1 RECORD TRUNCATED ON INPUT

A variable length record was longer than the value given by the /RECORD switch. The record has been truncated and the data lost.

?SRTSRS SPANNED RECORDS NOT SUPPORTED

IBM spanned EBCDIC records are not supported.

?SRTSSE SWITCH SCANNING ERROR

This is an internal SORT error. Send an SPR.

?SRTTMD TOO MANY DIGITS IN KEY

A COMP or COMP3 key was given with a length greater than 18 digits; or the key, if used with a FORTRAN binary file, was given with a length greater than 2 digits.

?SRTTMT TOO MANY TEMPORARY STRUCTURES SPECIFIED

More than the maximum number of temporary devices were specified. The maximum is 15. The devices specified after the fifteenth are ignored.

?SRTTNT TOPS-10 VERSION OF SORT WILL NOT RUN ON TOPS-20

Reassemble SORT with FTENEX==1.

SRTXPM EXPANDING TO nP

SORT uses its default memory algorithm to expand memory. This message notifies the user so that next time the sort can be run in a different size, if this size is not optimal.

The error codes in Table 5-2 are returned in AC after a RUN or GETSEG, in the right half of location E + 1 on 4-word argument blocks of LOOKUP, ENTER, and RENAME, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME.

Table 5-2
Error Codes

Symbol	Code	Explanation
ERFNF%	0	File not found, illegal filename (0,*), filenames do not match (UPDATE), or RENAME after a LOOKUP failed.
ERIPP%	1	UFD does not exist on specified file structures. (Incorrect project-programmer number.)
ERPRT%	2	Protection failure or directory full on DTA.
ERFBM%	3	File being modified (ENTER, RENAME).
ERAEF%	4	Already existing filename (RENAME) or different filename (ENTER after LOOKUP) or supersede (on a non-superseding ENTER).
ERISU%	5	Illegal sequence (RENAME with neither LOOKUP nor ENTER, or LOOKUP after ENTER).
ERTRN%	6	1. Transmission, device, or data error (RUN, GETSEG only). 2. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block. 3. Software-detected data inconsistency error detected while reading the UFD RIB or file RIB.
ERNSF%	7	Not a saved file (RUN, GETSEG only).
ERNEC%	10	Not enough core (RUN, GETSEG only).
ERDNA%	11	Device not available (RUN, GETSEG only).
ERNSD%	12	No such device (RUN, GETSEG only).
ERILU%	13	Illegal UUO (GETSEG only). No 2-register relocation capability.
ERNRM%	14	No room on this file structure or quota exceeded (overdrawn quota not considered).
ERWLK%	15	Write-lock error. Cannot write on file structure.
ERNET%	16	Not enough table space in free core of monitor.
ERPOA%	17	Partial allocation only.

Table 5-2 (Cont.)
Error Codes

Symbol	Code	Explanation
ERBNF%	20	Block not free on allocated position.
ERCSD%	21	Cannot supersede an existing directory (ENTER).
ERDNE%	22	Cannot delete a non-empty directory (RENAME).
ERSNF%	23	Sub-sirectory not found (some SFD in the specified path was not found).
ERSLE%	24	Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).
ERLVI%	25	Cannot create a SFD nested deeper than the maximum allowed level of nesting.
ERNCE%	26	No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK only).
ERSNS%	27	GETSEG from a locked low segment to a high segment which is not a dormant, active, or idle segment. (Segment not on the swapping space.)

APPENDIX A
SORT SWITCHES AT A GLANCE

Table A-1
SORT Switches

Switch	Position	Default
dev:output/UNLOAD dev:input/REWIND	Output Only Input Only	None
/INDUSTRY	Input or Output	
/KEY:F:L:O	Input and/or Output	None for /KEY:F:L but A for 0.
/ALPHANUMERIC /NUMERIC /COMPUTATIONAL /COMP1 /COMP3	After /KEY Switch	/ALPHANUMERIC (/NUMERIC if Sign Switch is given.) /SIGNED
/SIGNED /UNSIGNED	After Data Type Switch	/SIGNED if /NUM, /COMP, /COMP1 or /COMP3
/ASCII /SIXBIT /EBCDIC /BINARY	Input or Output	/ASCII if /ALPHA or /NUM /SIXBIT if /COMP or /COMP1 Fixed Length None
/FORTRAN /RANDOM /SEQUENTIAL		/SEQUENTIAL None
/BLOCK:n	Input and/or Output	Unblocked
/RECORD:n	Input and/or Output	None
/VARIABLE /FIXED	Input and/or Output	/FIXED if SIXBIT or EBCDIC /VARIABLE if ASCII
/LABEL:STANDARD /LABEL:NONSTANDARD /LABEL:OMITTED	Input and/or Output	Magtapes STANDARD Other devices OMITTED
/ALIGN	Output Only	Not word-aligned
/CORE:nP	Input or Output	Default Memory Algorithm

APPENDIX B
SCAN SWITCHES

The following is a list of optional switches to the SCAN program. They can appear in the command string with the SORT switches, but they are not needed to run SORT. These switches are preceded by a slash (/), and they may be abbreviated. If the user places a SCAN switch before a filename in the string, the switch will apply to all files specified. However, if the user places the switch after a filename, this switch will apply only to the filename that precedes it.

Table B-1
SCAN Switches

Switch	Meaning
/DENSITY:n	Sets tape density on input or output when reading a magnetic tape. n is either 200, 556, 800, or 1600 bpi. The default is installation dependent and is modified by the SET TAPE DENSITY (TO) n command. See the *DECSYSTEM-20 USER'S GUIDE.*
/HELP	Prints a message giving the general format of the command string and lists the switches available to the user.
/PARITY:EVEN or /PARITY:ODD	Specifies the parity to be used when reading a magnetic tape. The default is ODD.
/PHYSICAL	Suppresses logical device names for the specified device during LOOKUPS and ENTERS.
/PROTECTION:nnn	Gives the output file the protection code nnn (in octal).
/RUN:file specification	Runs the specified program after the sort is completed.
/RUNCORE:n	Sets the memory size of the program specified with the /RUN:file spec switch.
/RUNOFFSET:n	Runs the program specified with the /RUN:file spec with an offset of n. If the switch is omitted, the default is 0; if this switch is specified without a value for n, the default is 1.
/VERSION:n	Sets the output file version number to n.

APPENDIX C
CHARACTER COLLATING SEQUENCE

The following table gives the collating sequences for ASCII (DISPLAY-7) and SIXBIT (DISPLAY-6) fields as used in condition comparisons.

Table C-1
Character Collating Sequence

SIXBIT	Character	ASCII 7-Bit	SIXBIT	Character	ASCII 7-Bit	Character	ASCII 7-Bit
00	Space	040	40	@	100	`	140
01	!	041	41	A	101	a	141
02	"	042	42	B	102	b	142
03	#	043	43	C	103	c	143
04	$	044	44	D	104	d	144
05	%	045	45	E	105	e	145
06	&	046	46	F	106	f	146
07	'	047	47	G	107	g	147
10	(	050	50	H	110	h	150
11	)	051	51	I	111	i	151
12	*	052	52	J	112	j	152
13	+	053	53	K	113	k	153
14	'	054	54	L	114	l	154
15	-	055	55	M	115	m	155
16	.	056	56	N	116	n	156
17	/	057	57	O	117	o	157
20	0	060	60	P	120	p	160
21	1	061	61	Q	121	q	161
22	2	062	62	R	122	r	162
23	3	063	63	S	123	s	163
24	4	064	64	T	124	t	164
25	5	065	65	U	125	u	165
26	6	066	66	V	126	v	166
27	7	067	67	W	127	w	167
30	8	070	70	X	130	x	170
31	9	071	71	Y	131	y	171
32	:	072	72	Z	132	z	172
33	;	073	73	[	133	{	173
34	<	074	74	\	134	|	174
35	=	075	75	]	135	}	175
36	>	076	76	↑	136	~	176
37	?	077	77	←	137	Delete	177

APPENDIX D

EBCDIC CHARACTER SEQUENCE

Table D-1
EBCDIC Character Sequence

EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character
000	NULL	040	DS	100	Space	140	–
001		041	SOS	101		141	/
002		042	FS	102		142	
003		043		103		143	
004	PF	044	BYP	104		144	
005	HT	045	LF	105		145	
006	LC	046	EOB	106		146	
007	Delete	047	PRE	107		147	
010		050		110		150	
011		051		111		151	
012		052	SM	112	¢	152	
013		053		113	.	153	,
014		054		114	<	154	%
015		055		115	(	155	←
016		056		116	+	156	>
017		057		117	|	157	?
020		060		120	&	160	
021		061		121		161	
022		062		122		162	
023	TM	063		123		163	
024	RES	064	PN	124		164	
025	NL	065	RS	125		165	
026	BS	066	UC	126		166	
027	IL	067	EOT	127		167	
030		070		130		170	
031		071		131		171	
032	CC	072		132	!	172	:
033		073		133	$	173	#
034		074		134	*	174	@
035		075		135	)	175	'
036		076		136	;	176	=
037		077		137	¬	177	"

EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character	EBCDIC Code	EBCDIC Character
200		240		300		340	
201	a	241		301	A	341	
202	b	242	s	302	B	342	S
203	c	243	t	303	C	343	T
204	d	244	u	304	D	344	U
205	e	245	v	305	E	345	V
206	f	246	w	306	F	346	W
207	g	247	x	307	G	347	X
210	h	250	y	310	H	350	Y
211	i	251	z	311	I	351	Z
212		252		312		352	
213		253		313		353	
214		254		314		354	
215		255		315		355	
216		256		316		356	
217		257		317		357	
220		260		320		360	0
221	j	261		321	J	361	1
222	k	262		322	K	362	2
223	l	263		323	L	363	3
224	m	264		324	M	364	4
225	n	265		325	N	365	5
226	o	266		326	O	366	6
227	p	267		327	P	367	7
230	q	270		330	Q	370	8
231	r	271		331	R	371	9
232		272		332		372	
233		273		333		373	
234		274		334		374	
235		275		335		375	
236		276		336		376	
237		277		337		377	

APPENDIX E
ACCESSING ANOTHER USER'S FILE

The SORT program has two ways in which one user can access another user's file. The first way is via a logical name in place of the device name; the second is via a project-programmer number instead of a directory name. These methods are used in the SORT command line.

Project-programmer Numbers

> Should the user obtain a project-programmer number (i.e., a number similar to [4,204]) in an error message, he can use the TRANSL program to find out to which directory it corresponds. Refer to Section E.2.1.

For more information about referencing other user's files, refer to the DECsystem-20 User's Guide.

E.1 USING LOGICAL NAMES

To use a logical name in accessing another user's file, the user must do the following:

1. Give the DEFINE command to define a logical name (of no more than six characters) as the other user's directory name.

2. Use the logical name as the device name whenever giving the file specification.

E.1.1 Giving the DEFINE Command

To give the DEFINE command, the user:

1. Types DEF and presses the ESC key; the system prints INE (LOGICAL NAME).

   @DEFINE (LOGICAL NAME)

2. Types the logical name (ending it with a colon is optional), then presses the ESC key. The system prints (AS).

   @DEFINE (LOGICAL NAME) BAK: (AS)

3. Types the directory hame (enclosed in angle brackets) and presses the RETURN key. The system prints an @.

   @DEFINE (LOGICAL NAME) BAK: (AS) <BAKER>
   @

To check the logical name, the user can give the INFORMATION (ABOUT) LOGICAL–NAMES command.

```
@INFORMATION (ABOUT) LOGICAL–NAMES
BAK =><BAKER>
@
```

E.1.2 Using the Logical Name

The user includes the logical name in a command line.

E.1.2.1 Command Lines – To include the logical name in a command line, the user types the logical name in place of a device name.

The following example shows how the user would access the file <BAKER>NUMBER.SRT. (Remember that he has already defined the logical name BAK: as <BAKER>.)

```
@SORT
*NUMBER.UP=BAK:NUMBER.SRT/K:1:10/R:45
```

E.2 USING PROJECT–PROGRAMMER NUMBERS

To use a project-programmer number in accessing another user's file, the user:

1. Runs the TRANSL program to find the corresponding project-programmer number for the given directory name.

2. Includes the project-programmer number after the file type.

The user does not have to define a logical name when he uses a project-programmer number; project-programmer numbers, however, may not remain constant. Therefore, logical names should be used whereever possible.

E.2.1 Running the TRANSL Program

To run the TRANSL program, the user:

1. Types TRANSL and presses the RETURN key. The system prints TRANSLATE (DIRECTORY).

   ```
   @TRANSL
   TRANSLATE (DIRECTORY)
   ```

2. Types (or uses recognition on) the directory name and presses the RETURN key. The system prints the appropriate project-programmer number.

   ```
   @TRANSL
   TRANSLATE (DIRECTORY) BAKER
   <BAKER> IS [4,204]
   @
   ```

The user can also use the TRANSL program to make sure a project-programmer number is correct. He simply replaces the directory name with the project-programmer number.

```
@TRANSL
TRANSLATE (DIRECTORY) [4,204]
[4,204] IS <BAKER>
@
```

E.2.2 Using the Project-Programmer Number

The user includes the project-programmer number in a command line. Since project-programmer numbers may change, the user is advised to use a logical name.

E.2.2.1 Command Lines – To include a project-programmer number in a command line, the user types the project-programmer number after the file specification.

The following example shows how the user would access the file <BAKER>NUMBER.SRT. (Remember that he has already translated the directory name.)

```
@SORT
*NUMBER.UP=NUMBER.SRT[4,204]/K:1:10/R:45
```

GLOSSARY

ASCII	American Standard Code for Information Interchange. A 7-bit code in which textual information is recorded. The ASCII code can represent 128 distinct characters. These characters are upper and lower case letters, numbers, common punctuation marks, and special control characters.
Bias	The ratio of a Run to a Span. The bias should be approximately 2.0 if the file being sorted is truly random.
Buffer	A device or area used temporarily to hold information being transmitted between two processes such as external and internal storage devices.
Data	A general term used to denote any or all information (facts, numbers, letters, and symbols that refer to or describe an object, idea, condition, or situation). It represents basic elements of information which can be processed by a computer.
DSK	The generic device name for disk-like devices. It is translated by the system into actual file structure names which are defined for each job by the file structure search list.
EBCDIC	Extended Binary Coded Decimal Interchange Code. On this system, a 9-bit code intended for compatibility with other manufacturers' systems. Of the nine bits, eight are used for data.
File Specification	A list of identifiers which uniquely specify a particular file: name of the device where the file is stored, name of the file and the type, and the name of the directory.
Key	The part of the record being compared in order to sort the file into the correct sequence.
Label	On magnetic tape, it is the header or trailer record used to identify the reel. On DECtape and random-access devices, it is the directory block used by the system.
Low Segment	The segment of a user's virtual address space, beginning at zero.
Record	A collection of adjacent data items treated as a unit.
Run	The collection of records output in sequence to a work file.
Scratch Device	A device used to store intermediate-pass data during a sort.
SIXBIT	A 6-bit code in which textual information is recorded. It is a compressed form of the ASCII character set.
Sort	The results of comparisons between groups of records.
Span	The number of records that will fit in memory at any given time during a sort.
Stable Sort	A sort which leaves the relative order of equal keys unchanged.

INDEX

INDEX (Cont.)

SORT
User's Guide
DEC-20-USTGA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

If you require a written reply, please check here. ☐

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P.O. Box F
Maynard, Massachusetts 01754


digital