July 1978

This document discusses the features and uses of the SORT utility on TRAX systems. This document and the software to which it pertains are for use with RMS-11-formatted files only.

TRAX
SORT Reference Manual

Order No. AA-D346A-TC

OPERATING SYSTEMS AND VERSIONS: TRAX Version 1.0

SOFTWARE VERSION: 3.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation • maynard, massachusetts

First Printing, July 1978

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-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

CONTENTS (Cont.)

FIGURES

TABLES

Page

TABLES (Cont.)

PREFACE

This manual describes the features and operation of the SORT utility program under TRAX on the PDP-11. Chapter 1 describes the SORT program, its operation and environment. Chapter 2 is an operator's guide for running SORT. Chapter 3 is a reference for those who write the instructions for the SORT program. Chapter 4 describes error conditions, and Chapter 5 describes the use of files in SORT.

Appendix A lists the ASCII character set and a subset of the EBCDIC code. Table A-1 lists the ASCII code values in ascending octal sequence with the corresponding EBCDIC values. This is useful for comparing values when requesting a modified ASCII sorting sequence or when defining forced key fields. Table A-2 lists the EBCDIC subset used when you request an EBCDIC sorting sequence. The EBCDIC values are in ascending order with the corresponding ASCII values.

Appendix B contains two tables to help you determine the digit and zone values of printable characters. For example, the characters B, K, and S have an equal absolute digit value of 2; however, the "eleven" punch denotes a negative value. The digit values of J through R are negative (-) 1 through 9.

Appendix C gives examples of SORT applications.

Appendix D lists RMS status codes.

A glossary is included to define terminology used in this manual.

CHAPTER 1

INTRODUCTION TO SORT

The SORT utility program allows you to read any input file, sort the contents, and write out the sorted data onto an output file. The sorting sequence is determined by control fields, or key fields, within the data itself. If you do not wish to sort your data base, you can still use SORT to extract key information, sort that information, and store the sorted information on a permanent file. You can later use that file to access your data base in the order of the key information on the sorted file. The contents of the sorted file may be entire records, key fields, or record indices relative to the position of each record within the file (the first record on the data base is record 1, the second, 2, etc.).

SORT provides two sorting techniques:

1. Record Sort (SORTR)

   Record Sort produces a reordered file by using the entire contents of each record as the record key. A record sort can be used on any acceptable input device and can process any valid RMS (Record Management Services) format.

2. Tag Sort (SORTT)

   Tag Sort produces a reordered file by sorting only the record keys. SORT then randomly reaccesses the input file to create a resequenced output file according to those record keys. The tag sort method conserves temporary storage, but can only accept input files residing on disk.

The SORT utility program may be controlled by a command string and an optional specification file. There is a simple format for each. If your SORT application does not require that records be restructured or that only a subset of the input file be sorted, then you need only a command string to control SORT.

SORT can handle any RMS-valid file organization. Different file organizations are distinguished by the ordering of the records they contain and the way they handle the retrieval process.

The order of the records in a sequential file is determined by the order in which the records are read from the file. The first record in the file is the first record read out, regardless of whether the records are written to the file in some sorted order or not.

A relative file consists of record areas that are identified by relative record numbers. The first record area in the file is record number 1, the second is 2, etc., much the same as an apartment house where the first apartment is 1, and so forth. But, as in an apartment house, if you want the record that is in the twelfth record area, for

instance, you must ask for record number 12, even though there may not be records in areas 1 through 11. Relative files can reside only on disk.

An indexed file contains prologue information, one or more indices, and the data records themselves. To retrieve information, you ask for the proper record by primary or alternate key. The system looks up the key in the appropriate index and retrieves the record using the record pointer associated with the key. Indexed files can reside only on disk.

Table 1-1 shows the devices that you may use to supply data to SORT. Data may be stored in binary, ASCII, or EBCDIC form.

1.1 DATA FILES

SORT may accept a file from any one of the peripheral devices available in the system configuration:

1. Disk unit
2. Magtape unit
3. Support Environment terminal

A record is usually divided into several logical areas called data fields. The data in each field may or may not be relevant to SORT. Each field may be interpreted as a record identifier, key data, or general data related to the logical content of the record and not relevant to the sorting process. SORT uses record identifiers to distinguish the various types of records in a file. SORT uses the key fields in each record to reorder an input file. Any other data field in a record may be retained in the output file or ignored by SORT.

Figure 1-1 shows three different types of input records, each with a different format. The record identifiers are the letters in position 1: S means Sales record, O means Order record, and R means Restock record. In this case, the keys chosen for sorting the Sales record types are the "item number code" in positions 2 to 7, and the "number of items sold" in positions 8 to 13. The "total amount of sale" is an example of a data field not relevant to the sorting process.

Table 1-1
Selecting the Sorting Process and Devices
that Best Suit the Processing Environment

SORTING TECHNIQUE	INPUT FILE	OUTPUT FILE	WORK FILE
SORTR	Disk	Disk	Disk (3-8 files)
(Record Sort)	Magtape	Magtape*	
	Terminal	Terminal	
SORTT	Disk	Disk	
(Tag Sort)	Magtape	Magtape*	
		Printer	
		Terminal	

* Provided records are at least 18 bytes long. Magtape must be in ANSI format.

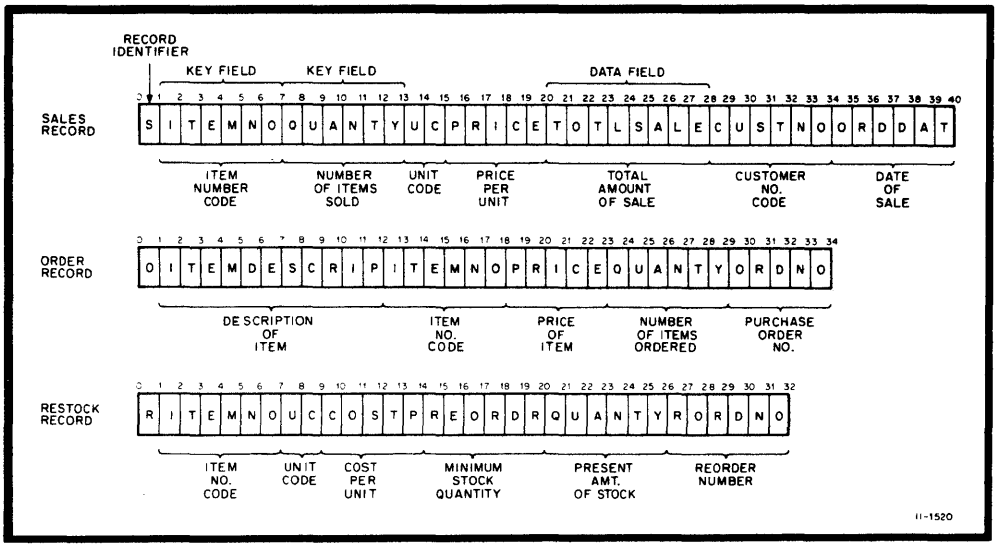


Figure 1-1 Sample Record Types

If you request a sort in ascending order on the sales records illustrated above, the sort is based on the item number code first and then on the number of each item sold within that item number. In order of decreasing significance, the keys are:

1. Item number
2. Number of items sold

The output file contains all sales records in the order shown in Table 1-2. Data fields other than those you use for keys, e.g., salesperson's code, may be in any order.

Table 1-2
Sorted Output File

First Key:Item Number	Second Key:Quantity
lowest item no.	lowest quantity
lowest item no.	next higher quantity
.	.
.	.
.	.
.	.
lowest item no.	highest quantity
next higher item no.	lowest quantity
next higher item no.	next higher quantity
.	.
.	.
.	.
.	.
next higher item no.	highest quantity
highest item no.	lowest quantity
highest item no.	next higher quantity
.	.
.	.
.	.
.	.
highest item no.	highest quantity

1.2 COMMAND STRING AND SPECIFICATION FILE

You invoke the SORT program by entering a command string. The command string has three functions:

1. To reference devices in the system for each file in the current sort.

2. To specify switches that define file parameters used in the sorting process.

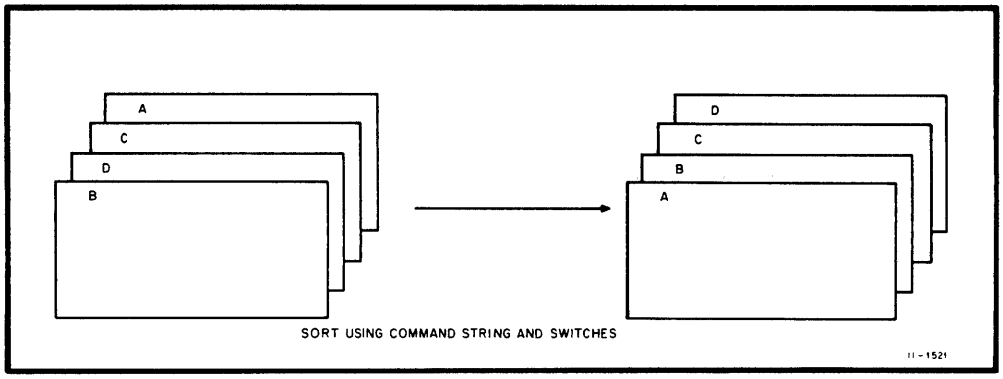
3. To reference a specification file or to specify other switches to control the sort.

Several command string switches define the sorting process parameters. One switch describes record formats and the maximum record size. Another delimits the internal work files. Others provide detailed file information to RMS.

Normally, you direct the sort with a specification file. Or, you may use two additional switches instead of a specification file to control a sort. One switch specifies the sorting process option; the other identifies the key fields. The use of these switches is limited to sorting an input file of uniform format:

1. The key fields must reside in the same location in every record of the input file.

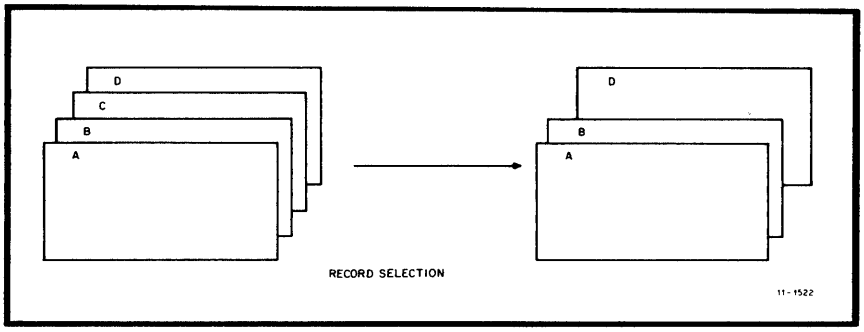
2. The file must contain only the records to be included in the sort. The figure below illustrates a general sort that would require only a command string and switches.



The specification file is the supplement to the command string, which provides the basis for controlling and directing the sorting process.

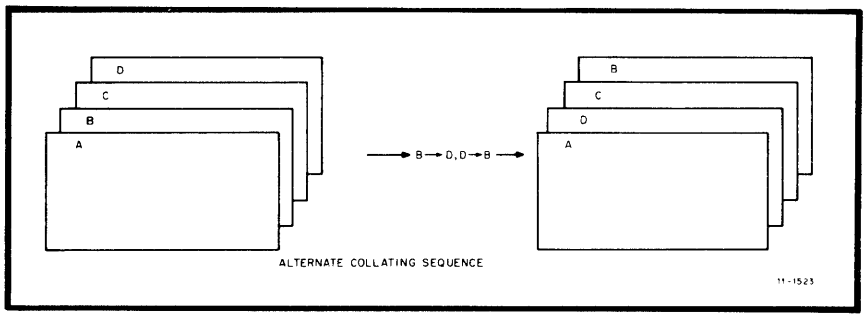
The specification file provides the following controlling features:

1. Record Selection



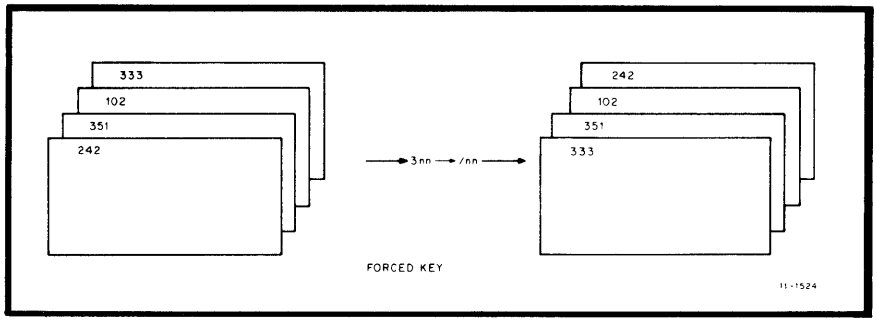
You can include or omit any records from the sorting process. The output file will contain only the records that you specify.

2. Alternate Collating Sequence



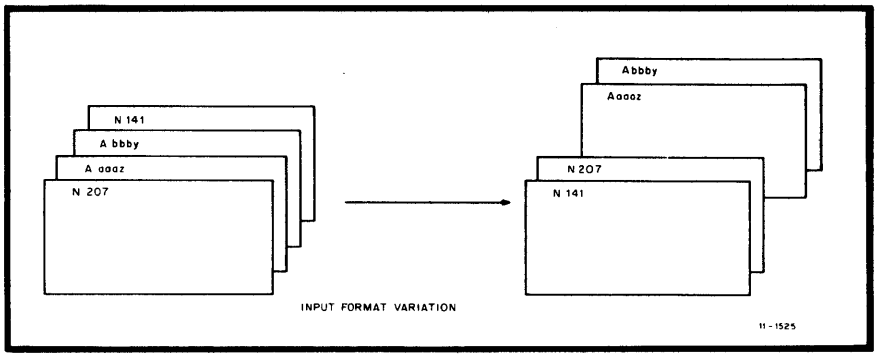
If necessary, you can specify an alternate collating sequence. The normal sequence is that implied in ASCII code. One alternate choice is EBCDIC values. The other is an individual alternate collating sequence (ALTSEQ). An ALTSEQ can be used to change the ASCII values of the normal sequence. It applies to all the alphanumeric key data in the records, but only during the actual sorting process. The output record remains unchanged.

3. Forced Keys



An ALTSEQ applies to all positions of the key. Forced keys allow you to specify an alternate sequence for particular positions within the key. You can specify an alternate sequence by substituting a lower-valued character, such as the slash (/) in the example above. Since the slash comes before 0, the 300-series records in the example are brought to the front of the file. Notice that the records so treated are in sequence and in front of the rest of the sorted file. The net effect is that of two sorted files, one behind the other.

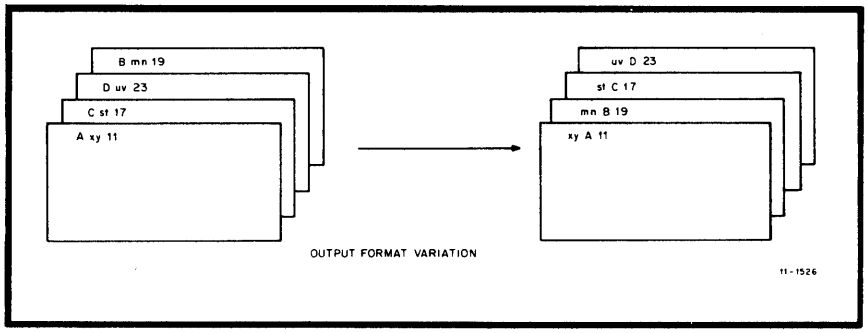
4. Input Format Variation



If the input file contains records with several different formats, you can identify those records by type so that they may be properly handled.

In the example above, A and N are record identifiers.

5. Output Format Variation



You can change the format of the data file during the sort, but you cannot change the contents of any given data item.

1.3 OPERATION OF SORT

The SORT program consists of two basic parts: a control program and a subroutine package called SORTS. The control program directs the overall processing. SORTS serves as a collection of subroutines that the control program uses during its processing.

There are three phases of operation in the SORT control program.* In the first phase, SORT reads the command string, decodes, and stores the switch values and the specification file, if present. Any errors in the command string or specification file are reported at this point.

* Each phase of operation corresponds to an overlay in the SORT program structure.

The second phase begins the pre-sort operation. The control program is called to open and read the input file and establish the keys. Then the SORTS subroutine begins the initial sorting process. At this point, the amount of available internal storage space becomes important to the efficiency of the sort. If that space is not sufficient to hold all the records, SORT builds strings of sorted records and transfers them to scratch files on bulk storage devices. In order to merge these files and complete the sort, space for at least three scratch files must be available. The SORT program normally provides for a maximum of eight scratch files. Either a switch in the command string or the amount of available internal work space can reduce the number of scratch files used.

The merge phase rebuilds the intermediate scratch files into a merged file. Another subroutine reads the records in the proper sequence. The records are then written into the output file. If there are no scratch files to merge because main memory was sufficient to hold all the records, the sorted records are written directly into the output file. After the last record is written, the control program cleans up the scratch files and returns to the first phase; SORT is then ready to accept another job.

1.4 SORT PROCESSING OPTIONS

There are two SORT processes: Record Sort and Tag Sort. You can specify the SORT process in the specification file or with a switch in the command string. The default process is the Record Sort. Each process has its particular input requirements, processing methods, and resultant output files.

1.4.1 Record Sort (SORTR)

The Record Sort (SORTR) outputs all specified record data in a sorted sequence. Each record is kept intact throughout the entire sorting process. Since it moves the whole record, SORTR is relatively slow and may require considerable main memory or external storage work space for large files.

NOTE

Records can be FIXED, STREAM ASCII, or VARIABLE length in any valid RMS (Record Management Services) format. The size of the records on a fixed length format file is determined when the file is created. Variable length records carry a count along with them that specifies their length. This length data is used by the RMS routines and is transparent to SORT.

Table 1-3
Sorting Process Options

Type of SORT	Type of File	Record Size and Format	Device	Speed
SORTR (RECORD SORT)	Input and Output	any	Any appropriate device including: disk, magtape*	Slow
SORTT (TAG SORT)	Input	any	disk	Slow for large files, large keys
	Output	any	any appropriate device (including magtape*)	

* Magtape labels are ANSI standard.

1.4.2 Tag Sort (SORTT)

The Tag Sort (SORTT) produces the same kind of output file as SORTR, but it only handles record pointers and key fields. Since SORTT moves a smaller amount of data than SORTR, SORTT usually performs a faster sort than SORTR. The input file must be randomly re-accessed to create the entire output file, which may be a lengthy process for large files.

CHAPTER 2

USING THE SORT COMMAND

The SORT utility is installed as part of the TRAX Support Environment. It can be invoked from any TRAX support terminal by using the SORT command string. This chapter describes the available forms of the SORT command. See Chapter 3 of this manual for details about specific properties of the SORT utility.

2.1 THE SORT COMMAND STRING

Two forms of the SORT command are possible. The standard form uses the syntax:

```
>SORT[/Qualifier(s)]  Input-Filespec[/file-qualifier(s)]
```

Qualifiers and File-qualifiers are discussed in Chapter 3. They are used to specify the type of sort and the attributes of the file being sorted, as well as passing control information to the SORT utility.

A second form of the SORT command allows for interactive prompting of the input file specifications:

```
>SORT[/Qualifier(s)]
FILE?Input Filespec[/file-qualifier(s)]
```

To illustrate the SORT command string, consider a payroll program example. The payroll program has generated a disk-based file of 80-character fixed-length records called WAGES.DAT which contains the Social Security Account Number along with wage and tax data for each employee of a firm. Prior to reporting wages to the Social Security Administration, you must verify that the file is in ascending SSAN sequence. The following command string will sort the file WAGES.DAT on a nine-character SSAN key beginning in character position one of the wage record.

```
>SORT/KEYS:(1.9)/PROCESS:RECORD WAGES.DAT/FORMAT:FIXED:80
```

The qualifiers /KEYS and /PROCESS, as well as the file-qualifier /FORMAT are explained in detail in Chapter 3.

After you enter the SORT command string, the SORT utility begins processing. Any errors detected in the command string or specification file are printed immediately on your support terminal. At the end of a successful run, SORT returns the following messages:

```
SRT -- M:ELAPSED REAL TIME:  hh:mm:ss
SRT -- M:TOTAL RECORDS SORTED:  nn
```

This message reports the wall-clock time necessary to perform the sort and the number of records in the output file. You are then returned to the TRAX operating system where you may enter additional SORT command strings, or go on to some other operation.

2.1.1 Multiple Line Input

In the event that your SORT command string becomes too long to fit in a single line, division of the command string is possible. Simply use a hyphen (-) as the last printing character of a line, enter a carriage return, and continue your command string on the next input line. You should avoid splitting qualifiers and their arguments over more than one line.

Referring again to the example of the wages file sort, you could divide that command string as follows:

```
>SORT/KEYS(1.9)-
DCL>/PROCESS:RECORD -
DCL>/WAGES.DAT/FORMAT:FIXED:80
```

DCL> is the prompt of the DEC Command Language interpreter, which decodes your command string and passes this information to the operating system. Note that the second line PROCESS:RECORD - has a space after the word RECORD. This space is required to distinguish the command qualifiers from the input file specification and should be present at all times. You may also subdivide indirect command file input strings in the manner described above.

2.2 INDIRECT COMMAND FILES

The TRAX Support Environment allows you to create indirect command files which reside on your system device and are made up of command string input. With the SORT command, this facility allows you to create a command file which can then be executed at regular intervals for certain recurring types of processing. In the case of the wages example cited earlier in this chapter, you might create a file WAGSRT.CMD which contains the following information:

```
SORT/KEYS:(1.9)/PROCESS:RECORD  WAGES.DAT/FORMAT:FIXED:80
```

You execute this indirect command file by typing the following string:

```
>@WAGSRT
```

The file WAGSRT.CMD is then retrieved by the indirect command file processor and decoded to invoke the SORT command with the qualifiers you have specified.

2.3 RMS-11 FILE SPECIFICATION INFORMATION

A TRAX file specification conforms to standard RMS-11 conventions. It has the following form:

```
device:[ufd]filename.filetype;version
```

where:

device is the name of the physical device on which the volume containing the desired file is mounted. The name consists of two ASCII characters followed by a 1- or 2-digit octal unit number and a colon(:); for example, LP0: or DB1:. A logical device name may also be used.

Dev:	Device
SY:	system disk, default device
TI:	user's terminal
TTn:	Support Environment Terminal
LPn:*	line printer
DBn:	RP04/05/06 disk
DMn:	RK06/07 disk
DRn:	RM02/03
MMn:	magnetic tape (TU45/TE16)

* n specifies the device unit number; if n is not specified, n is assumed to be 0.

[ufd] is the user file directory specification consisting of two octal numbers in the range of 1 through 377 (octal). These numbers must be enclosed in brackets and separated by a comma and must be in the following format:

```
[group,member]
```

For example, member 225 of group 300 would use the following entry:

```
[300,225]
```

filename is the name of the desired file. The file name can be from one to nine alphanumeric characters; for example, BILLRTN. You must always specify the file name. There is no default specification for this component. Failure to specify the file name causes an error to be generated.

filetype is the 3-character file type identification. Separate the file name and filetype with a period (.). Files with the same name but different functions are distinguished from one another by the filetype; for example, BILL.TSK and BILL.OBJ might be the task file and the object file, respectively, for the program BILL. You may omit the file type, but you should avoid this practice when dealing with system programs which generally assume a default filetype for various operations. The wild card (*) specifier may be used in place of a file type. The action specified will be applied to all file types associated with a given file name or wild card file name. (e.g., *.* specifies the current version of all files in the user file directory.)

version is the octal version number of the file in the range 1 through 77777 (octal). Separate the extension and version by a semicolon (;). Various versions of the same file are distinguished from each other by the version number; for example, BLL.OBJ;1 and BILL.OBJ;2 are successive versions of the same file. The version number may be omitted, in which case the current (highest-numbered) version is assumed. To act upon all versions of a file, you can use the wild card (*) specification.

The device, user file directory code, filetype, and the version specification are all optional.

Table 2-1 lists the default assumptions applied to missing components of a file specification. Table 2-2 lists the filetypes assumed by TRAX system software.

Table 2-1

Item	Default
device	user's current default device
[ufd]	user's current default [ufd]
version	for an input file, the default version number is the existing version with the highest (octal) number for that file. for an output file, the default is calculated as one greater than the highest existing version number for that file.

Table 2-2

File Contents Description	Default Filetype
task image file	.TSK
memory allocation file	.MAP
symbol definition file	.STB
object module	.OBJ
object module library file	.OLB
overlay description file	.ODL
indirect command file	.CMD
COBOL source text file	.CBL
BASIC-PLUS-2 source file	.B2S
MACRO-11 source file	.MAC

CHAPTER 3

SORT EXECUTION OPTIONS

3.1 SWITCHES

SORT switches describe the input and output files and allow you to change SORT defaults, such as the number of scratch files used. All numeric switch values are decimal, not octal. All switches are valid for both input and output files unless otherwise noted. Table 3-1 summarizes the switches and other pertinent information for your convenience.

3.1.1 ALLOCATION (/AL:n)

The ALLOCATION switch is valid for the output file only. You can use this switch to specify the initial space allocation for the output file. Legal values range from 0 to 65,535. If you do not use this switch, the output file allocation defaults to the input file size for the record and tag sort processes.

3.1.2 BLOCKSIZE (/BL:n)

The BLOCKSIZE switch is valid for magtape files only and is used to specify a nonstandard magtape blocksize. If this switch is not used, the blocksize defaults to the standard 512-byte block.

3.1.3 BUCKETSIZE (/BU:n)

The BUCKETSIZE switch specifies the RMS bucket size (the number of 512-byte blocks per bucket). Please note that the size of the block is the standard 512 bytes even if the BLOCKSIZE switch is used. The default value of this switch depends on the organization of the input file. If the input and output files are of the same organization, the default for the output file is the input file value. If the input files differ in organization or the default is required for the input file, the default value is 1.

Table 3-1
Switch Summary

Switch Name	Code	Default Value	Validity
ALLOCATION	/AL:n	See Section 3.1.1	Output
BLOCKSIZE	/BL:n	512 bytes	Both (magtape)
BUCKETSIZE	/BU:n	Input file or 1	Output
CONTIGUOUS	/CO	Non-contiguous	Output
DEVICE	/DE:x	SY:	Input
FILES	/FI:n	5	Both
FORMAT	/FO:x:n	(See Section 3.1.7)	Both
INDEXED SEQUENTIAL	/IN:x	(See footnote)	Input
KEY	/KE:abm.n	C, N, first position of field, length of field	Input
OUTPUT	/OU:filespec	Input file	Output
PROCESS	/PR:x	R (record sort)	Input
RELATIVE	/RE	(See footnote)	Output
SEQUENTIAL	/SE	(See footnote)	Output
SIZE	/SI:n	(See Section 3.1.14)	Output

* The default for input files is the type of file present at OPEN time.

The default for output files is SEQUENTIAL. The output record format will default to the characteristics of the input file unless you use STREAM ASCII input record format. STREAM ASCII will produce a VARIABLE output record format.

Default record sizes are the size of the input record.

3.1.4 CONTIGUOUS (/CO)

The CONTIGUOUS switch is valid for output files only. It specifies that the output file will be contiguously allocated. This means that each successive block assigned to a file will be physically located between its logical predecessor and its logical successor with no filler or extraneous material separating the blocks. The default is non-contiguous.

3.1.5 DEVICE (/DE:x)

The DEVICE switch lets you specify which device you want to be associated with the scratch files. This switch overrides any device specification from task build options. The parameter x is any valid 1- to 4-character device name with the colon that normally follows that name omitted (e.g. DB3).

3.1.6 FILES (/FI:n)

The FILES switch specifies the maximum number of intermediate scratch files. The default is 5.

3.1.7 FORMAT (/FO:x:n)

The FORMAT switch specifies the record format (x) and the maximum record size (n) in a file. The record format x may take the following values:

FIXED,
STREAM,
VARIABLE, or
UNKNOWN

Record format values may be truncated to the first letter. The maximum record size n is the exact record size in bytes for FIXED length records. For the other formats, n is the size in bytes of the largest record present. The record size is required only on input.

This switch must be present in input files even if the format is UNKNOWN. The default of the record format (x) is VARIABLE. The record format on output can be used to override the input record format. For the default of n, see the footnote of Table 3-1.

3.1.8 INDEXED SEQUENTIAL (/IN:X)

The INDEXED SEQUENTIAL switch specifies the INDEXED SEQUENTIAL file organization. For default values, see the footnote of Table 3-1. X specifies the number of keys (no default). This switch is valid on input only.

3.1.9 KEY (/KE:abm.n)

The KEY switch is the most important switch in the SORT parameters. It tells SORT which fields are to control the sequence of the output file. You can specify more than one key field (up to 10) if you separate each description from the next with a colon. Here is an example of two key fields:

```
/KE:BN1.6:C8.2
```

The field description sequence in the basic format above is abm.n which breaks down as follows:

a — specifies the way in which the data is to be handled. If it is omitted, the default value is C as described below. But first, if you have not used SORT before, take a look at Appendix B and become acquainted with the concepts of zones and digits.

Here are the legal values for the data handling parameter and their interpretations:

B — two's complement binary

C — alphanumeric

D 1. if the characters are alphabetic, numeric with the sign superimposed over the units digit, or contain slashes (/), use the value of the digits group (see Table B-1). Here are two examples and their values:

A2CD5 = (+)12345 A/47J = (-)11471

If the characters represent a standard number, such as 12, -35, 42.98, or -0.76E+3, convert the number to binary for storage or evaluation

F 2- or 4-word floating point binary

I same as D, but with the sign leading and separate, so that the first byte of the field is a + or -

J same as I but with the sign trailing and separate

K same as D but with the sign leading and overpunched (54321, for instance, if positive, would come out as 5432A. The negative of 54321 would be 5432J.)

P packed decimal format

Z ASCII zone (see Table B-2)

b defines the general sort order. The default is N (ascending order).

N ascending order

O opposite, or descending, order

m is a decimal number giving the first byte of the key field. Number from the first byte of the record which is byte 1. This item must be present.

n is a decimal number giving the length of the key field in bytes. This item must be present.

3.1.10 OUTPUT (/OU:filespec)

The OUTPUT switch specifies the file that will receive the sorted records. Default is the input file.

3.1.11 PROCESS (/PR:x)

The PROCESS switch specifies the type of sorting process to use. Legal values are:

R - RECORD (the default)
T - TAG

3.1.12 RELATIVE (/RE)

The RELATIVE switch, for output only, specifies RELATIVE file organization. For defaults, see the footnote of Table 3-1.

3.1.13 SEQUENTIAL (/SE)

The SEQUENTIAL switch, for output only, specifies SEQUENTIAL file organization. For defaults, see the footnote of Table 3-1.

3.1.14 SIZE (/SI:n)

The SIZE switch specifies the size of the retrieval window. The range and default for the switch are determined by your system manager.

3.2 THE SPECIFICATION FILE

The specification file offers a variety of controls for the sorting process. These controls are entered in a format consisting of up to three types of records or lines in the specification file; a fourth type may be used if nonstandard collation is required. Figure 3-1 shows two possible arrangements of the specification file whose elements are explained below.

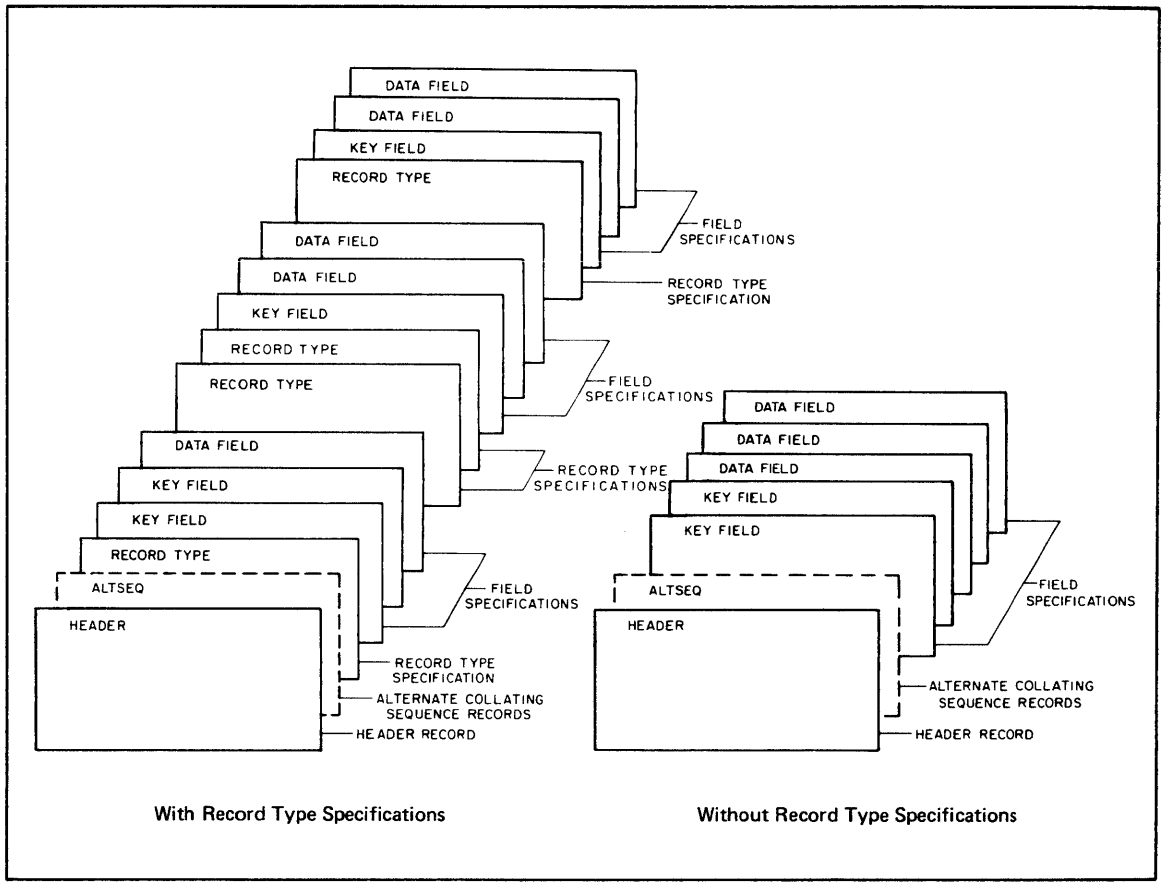


Figure 3-1 Specification File Format

1. The Header

   The first record in a specification file must be the Header. The header tells the SORT program:

   - the kind of sorting process to be used,
   - the key field and output record size,
   - whether an alternate collating sequence is to be used, and
   - whether the key fields are to be stripped from the output. There is only one Header for each SORT execution.

2. ALTSEQ Records

   If you use an alternate collating sequence, ALTSEQ records follow the header. ALTSEQ records allow you to change the order in which a character or characters is sorted. You can use these records to:

   1. Move certain records to the front or back of a file
   2. Group them in one place within the file
   3. Change the order in which records appear in a given sequence within the file

3. Record Type Specifications

   If you do not use an alternate collating sequence, the next type of record or line in the specification file is record type specifications. They control record selection and allow SORT to work with variable record formats. If all records have an identical format (or the format is not important) and all are to be included in the output file, the record type specification may be omitted.

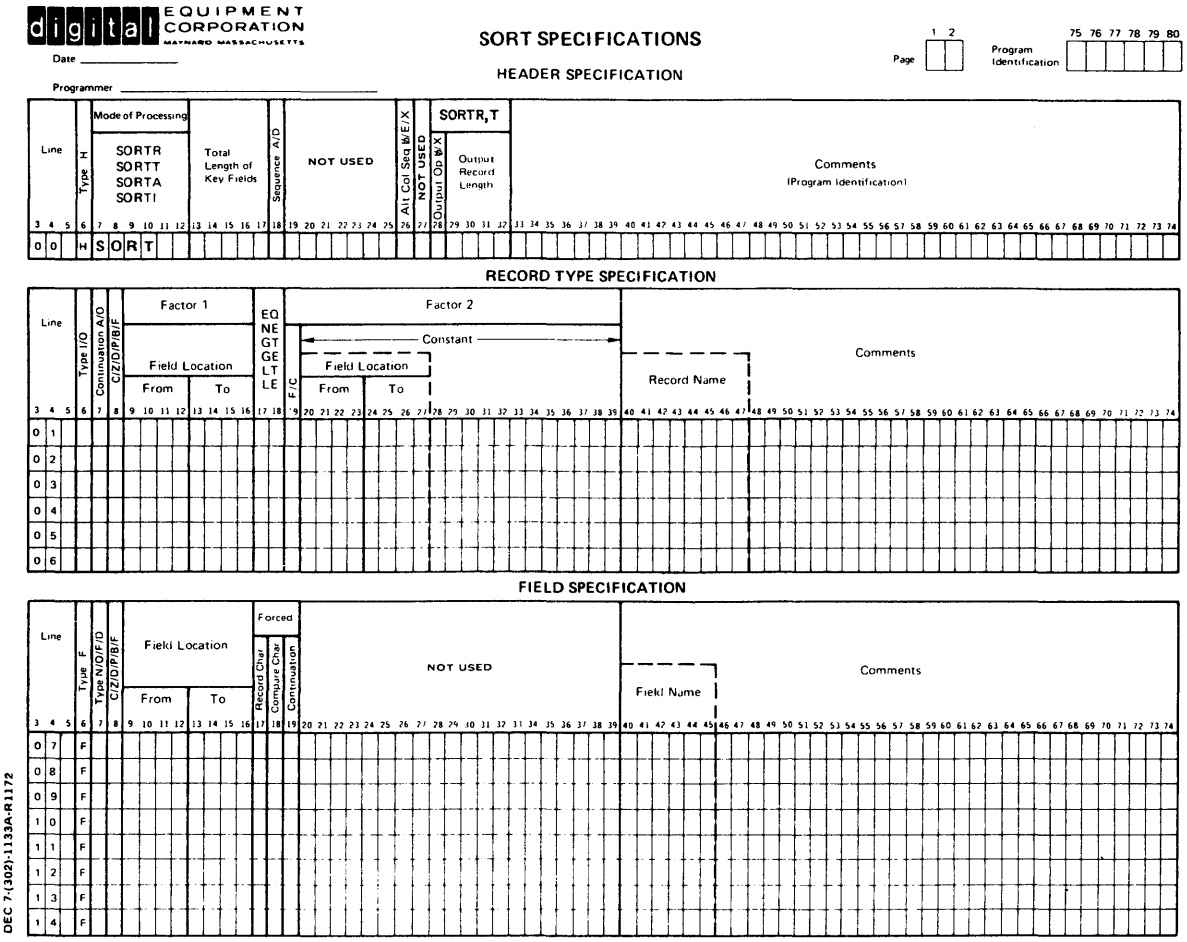
4. Field Specifications

   If record type specifications appear, each is followed by any applicable field specifications. If not, the field specifications follow the header (behind any ALTSEQs). The two kinds of field specifications are key field specifications and data field specifications. Key field specifications define and locate each key. They are listed in order of decreasing significance. Data field specifications define and locate all the data fields to be written to the output file. The data will appear in the same order in the output file as the data field specifications.

   Key field specifications appear first in the specification file and are followed by any data field specifications.

   The format of each type of specification record is fixed. The SORT specification form, is shown in Figure 3-2. The following entries are common to all three types of specification file lines.

Column	Entry	Notes
1-2	Page number	Required only when different types of records are to be described. A separate page, numbered in ascending sequence, should be used for each record type and its corresponding Field Specifications. Only the first page has a header specification.
3-5	Line number	Depicting line sequence. If column 5 is blank, 0 is assumed. Thus a digit entry in this column can be used to identify later line insertions.
6	Specification Type	H, I, O, or F as shown below
40, etc.	Comments	Ignored by SORT processing



DEC 7-(302)-1133A-R1172

Figure 3-2 SORT Specification Form

NOTE

Whole lines of comments may also be included in the file; these are denoted by an asterisk (*) in column 7. The comment lines may appear anywhere in the specification file and serve to document the file in its printed listings. They do not affect SORT operations.

In the following material, unless otherwise stated, the following criteria apply:

- Numeric data is decimal, not octal.
- Either leading zeroes or leading blanks are acceptable in right-justified entries.
- All field position definition records begin at column 1.

3.2.1 Header Specification

See Section 3.2.1.1 for notes and comments:

Columns	Explanation and Legal Entries
1 - 5	Page and line numbers: described in Section 3.2
6	Header record identification Legal value: H
7 - 12	Type of SORT (must be left-justified) Legal values: SORTR or blanks - Record SORT SORTT - Tag SORT
13-17	Total of all key field sizes Legal values: 1 - 16383 N.B.: must be equal to the total size in bytes of the largest record key on the file and right-justified
18	Normal sort order seguence Legal values: A or blank - ascending D - descending
19 - 25	(not used)

26 Alternate collating sequence

Legal values:

blank - standard ASCII sequence
E - EBCDIC sequence
X - user-modified ASCII sequence

27 (not used)

28 Output option

Legal values:

X - key fields created by SORT will be stripped from the output record
blank - no action

29-32 Output record length

Legal values:

decimal number equal to the number of bytes in the largest output record

33-39 (not used by SORT - may be used for comments)

3.2.1.1 Notes and Comments on Header Specification Entries

Column of Entry Affected	Notes and Comments
18	This field may be qualified by N or O entered in column 7 of the field specification (q.v.)
26	This field identifies the sort order for fields identified as alphanumeric (C in column 8 of the field specification). If X is entered here, the header must immediately be followed by ALTSEQ cards. See Section 3.2.1.2 for ALTSEQ card format. Standard ASCII is assumed for forced key field specifications and this field cannot be used to amend that assumption. SORT does not perform file transliteration. **Files that are not already in standard ASCII characters must be transliterated before the sort.** The EBCDIC option specifies ASCII characters sorted into EBCDIC sequence.

28	If a blank appears here, the key fields may be treated as data fields and moved to the output record as any other data field would be, whether relocated within the output record with respect to the input record or not.
29 - 32	To determine this number, add the sizes of all the data fields in the field specifications (plus the sizes of the key fields if column 28 is blank) for the largest record in the file. If neither SORTT nor SORTR is to be run at this time, then an entry in this field is not needed.

3.2.1.2 ALTSEQ Format and Notes

Column	Notes
1 - 6	ALTSEQ - mandatory entry
7 - 8	(not used)
9 - 80	Replacement character groups (12), of 6 positions each; each group contains the octal representations of two ASCII characters: the character being sorted out of sequence, and the character that corresponds to its new position in the sorting sequence. For example, the correct entry to replace an ASCII space with an ASCII zero (0) would be 040060; 040 for the space and 060 for the zero. The result of this would be to sort spaces into the same relative position in the sequence as 0s. Notice that preexisting 0s are not affected by this entry, but they may be affected by another entry on the same or another ALTSEQ card under the same header.

3.2.2 Record Type Specification

See Section 3.2.2.1 for notes and comments.

Columns	Explanation and Legal Entries
1 - 5	Page and line numbers. See Section 3.2 for description
6	Inclusion or Omission character Legal values: I - Include the records described in this line in the sort O - omit the records described in this line from the sort

7 — Continuation character (allows for cases in which several conditions define a single record type)

Legal values:

A - logical AND
O - logical INCLUSIVE OR

8 — Field mode (defines the processing to be applied to the data field defined)

Legal values:

B - binary
C - alphanumeric
D - numeric
F - binary floating point in 2 to 4 words
I - numeric with leading and separate sign
J - numeric with trailing and separate sign
K - numeric with sign leading and overpunched in first byte
P - packed decimal
Z - zone, see Table B-2, Appendix B

9 - 12 — Factor 1 (position of the base comparison factor is in the first byte; the first byte in the record is byte 1)

Legal values:

a decimal number - location of the first byte

blanks - specifies a single byte field

13 - 16 — Factor 1 endpoint

Legal values:

a decimal number - location of last or only byte in the field

17 - 18 — Relationship of Factor 1 to Factor 2

Legal values:

EQ - equal
NE - not equal
LT - less than (i.e. F1 LT F2)
GT - greater than
LE - less than or equal to
GE - greater than or equal to

19	Comparison factor format (indicates whether Factor 2 is a field or a constant) Legal values: F - identifies a field C - identifies a constant
20 - 39	Factor 2 (This takes two different formats depending on whether Factor 2 is a field or a constant.) If Factor 2 is a field: 20 - 23 location of first byte - a decimal number or blanks as for Factor 1. 24 - 27 location of the last or only byte - a decimal number as for Factor 1. 28 - 39 (not used) If Factor 2 is a constant: 20 - 39 the value of the constant
40 - 80	(not used - available for comments)

3.2.2.1 Notes and Comments on Record Type Specifications

Columns	Explanation
(general)	If all the records in the input file are to be included in the sorted output file and they all have the same format or you are using the entire record as the key, then no record type specification is necessary. If all the records have the same format and some are to be omitted, then only one record type specification is needed. If several different formats of records are to be sorted, then there must be a record type specification for each. Each record type specification used should be followed by its appropriate field specifications.
6	If this column contains an I and there are no further entries in this line, all records not previously described are to be included in the sort input. **Records that are not described in Include lines will be ignored.** Because SORT processes Include and Omit lines sequentially, you should be very careful to get your lines in the right order, particularly if any records are described on more than one line. The last Record Specification line before a Field Specification must be an Include line. If you describe the records to be processed by omitting the unwanted records from the file, then you can use a line with an I in column 6 and no further entries in the line as the final record specification.

7 Multiple AND and OR lines are permissible, but the series will be processed strictly in sequence, each line ANDed or ORed to the accumulated Boolean expression in turn:

If A is False and B and C are True, the expression A OR (B AND C) is True logically, but in order to get the correct results from SORT, arrange it this way:

B AND C OR A

SORT will react:

(B AND C) OR A,

which gives the required True value.

On the other hand,

A OR B AND C

gives a True value in both the SORT and non-computerized logic as it stands.

8 C - alphanumeric information limited to 256 bytes

D - numeric value, maximum size 8 bytes after conversion to binary. If the number is presented in standard ASCII numeric format, as for a decimal number with or without decimal point, floating point exponent, or sign at the beginning, the SORT will convert the number to binary. But the maximum size is still 8 bytes.

F - maximum size 8 bytes - two or four word floating binary representation before conversion

Z - ASCII zone, maximum size 1 byte - see Table B-2 in Appendix B

9 - 16 Factor 1 - Sort identifies records by comparing the value of one or more fields in a record with constants or other fields in the record. These other fields or constants are referred to as Factor 2.

17 - 18 defines the Factor 1 - Factor 2 relationship

20 - 39 Factor 2 - See notes under Factor 1 above. If Factor 2 is a field, it must be the same length as Factor 1. If a constant, after conversion to the corresponding internal format, be the same length as Factor 1 and be in the same mode (see Field Mode, position 8).

3.2.3 Field Specifications

There are normally a set of field specifications following each record type specification in the specification file set. (For one exception, see paragraph 3.2.2.1, general notes.) If there are no record type specifications, any appropriate field specifications follow the header. Field specifications have two main jobs:

- to describe the output file format
- to determine, for each record type, the keys on which the sort is to be based.

List the key field specifications first, in order of decreasing significance. Then add data field specifications to put out any portions of the original input record.

List the Key and Data field specification lines in the specification file in the same order as the required output record format. SORT creates the output record according to the order, size, and contents of the key and data specification lines. Place an X in column 28 of the Header to strip unwanted key fields from the output record. If the key field is specified twice, as a key field and as a data field, the key field can remain in the output record where the data field specification puts it, even if the key fields are stripped. See Section 3.2.3.1 for further commentary.

Columns	Explanations and Legal Entries
1 - 5	Page and line numbers - see Section 3.2
6	F - specifies a field specification
7	Field type - specifies keys and their sort sequences and data fields Legal values: N - key field, normal sort sequence (see position 18 of Header) O - key field opposite sequence F - key field, special treatment required (see positions 17 - 19) D - data field (if used, then column 8 should be C)
8	Field type - should be C if column 7 is D - these options are the same as those for column 8 of the record type specifications
9 - 12	Field location - location of first byte of multi-byte field Legal values: a decimal number - location of first byte of field blanks - specifies one-byte field

13 - 16 Field location - last or only byte

Legal values:

a decimal number - location of last or only byte in the field

17 - 19 Forced key fields

17 Trigger character - identifies the character needed to trigger the force:

Legal values:

blank - the force always occurs
any other character - the force occurs only if this character is found

18 Replacement character

Legal values:

any legal character

19 Continuation character

Legal values:

blank - this is a new forced key

any other character - this key is continued from the previous line

20 - 80 (not used - available for comments)

3.2.3.1 Notes and Comments on Field Specification Entries

Columns	Explanation
7	Enter key fields first in order of decreasing significance.
17 - 19	Use this field to change the content of a 1-character key field, thereby changing the sorting sequence. The original data will still appear on the output record, since only the key will be changed. To specify a forced key, put F in column 7 of the field specification and the nature of the force in columns 17 - 19. Since the force is based on characters, there must be a C in column 8. A force does not apply to any other keys or fields than the one stated.

To help you understand how these forced key fields affect the SORT, here is a description of how the SORT handles these fields:

When column 7 contains an F, SORT sets up a one-byte field in its key area and sets its value to either 377(octal) for ascending or 0 for descending sequence. Note that the alternative collating sequence does not apply to forced key characters.

If column 17 is not blank, SORT compares its contents to the content of the field defined in columns 13 - 16. If an equal condition occurs, SORT moves the character to the reserved key byte previously described.

By using more than one Field Specification, you can force more than one character out of normal sequence, such as sorting the entire alphabet out ahead of numerics. But beware:

1. Unless the displaced character is itself displaced to another position in the sorting sequence, it will appear with the character displacing it in the order in which the two characters were found in the input file. Thus, if A displaces ! and a series appears in the input file as (A, B, !, A, !, !, A, C, D), the sorted output will be (A, !, A, !, !, A, B, C, D), so...

2. When in doubt, force the displaced character elsewhere, particularly when it is known to appear in the input file and is not omitted elsewhere in the SORT specification file.

If, on the other hand, column 17 contains a blank, no comparison is performed and the value of column 18 always replaces the initial value in the reserved key byte. This has the following results:

1. When associated with conditional keys in preceding lines, records not satisfying the prescribed conditions can be grouped into any position in the output file, not necessarily the end of the file.

2. When this specification stands alone, it becomes an unconditional force of all records in the group. This form can be useful when the group is already identified by a record type specification. It has no effect on the entire file if applied to the entire file.

If column 19 of the field specification is blank or the preceding line does not specify a force key, SORT only reserves a new key byte. The same byte is modified in all other cases. This makes it possible to combine several conditions within the same key, such as arranging an output file in the order 4, 2, 3, 1, rather than 1, 2, 3, 4 or 4, 3, 2, 1. However, SORT still expects the field location to be defined in columns 13 - 16, even if each continuation line covers the same field.

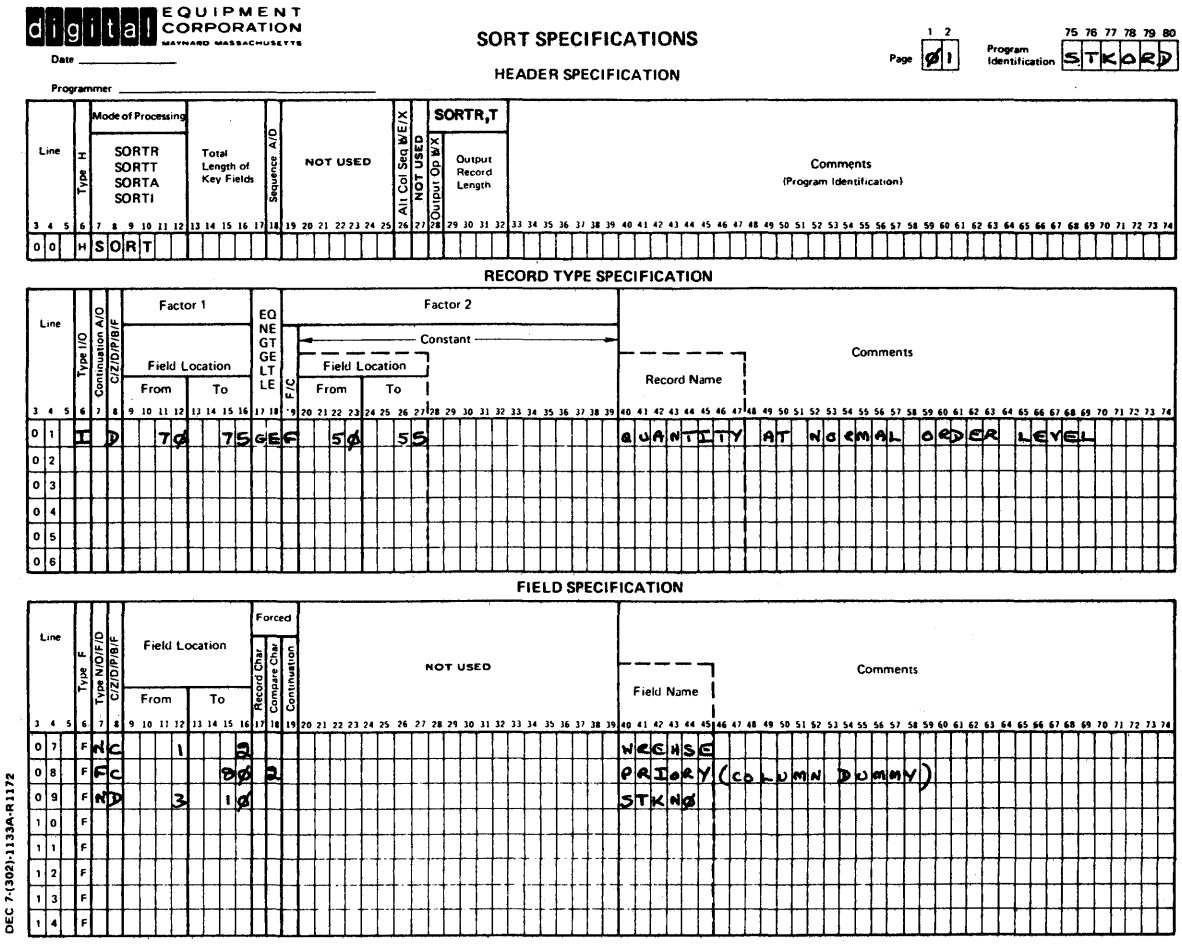
3.2.4 Using a Specification File

The sample input file contains records in the format shown in Figure 1-1. The file may contain null records such that there may be a record identifier and no other data in the record.

SORTR is entered in columns 7 through 11 of the Header. The header specification also indicates that the key is to be sorted in ascending sequence and that an alternate collating sequence is to be used. The ALTSEQ line following the header tells SORT to interpret all nulls as spaces.

The null records are omitted from the sorting process with an Omit line in the record type specification. The Include lines direct SORT to process only the RESTOCK records where the present amount of stock is less than the stock quantity allowed. ORDER Records, SALES Records, and RESTOCK records where the present amount of stock is greater than the stock quantity allowed are ignored.

The field specification shows that the selected records will be sorted on the reorder number. Since this is a SORTR, the Item Number Code, Unit Code, and Cost per Unit data fields, in positions 2-14, transfer to the output records. The output records will have the reorder number in positions 1-6, and the Item Number Code, Unit Code, and Cost per Unit in positions 7-19. Note that the total length of the key fields and the output record length in the header specification are calculated from the field sizes indicated in the field specification.



DEC 7-(302)-1133A-R1172

Figure 3-3 Header Specification Entries

3.3 SORT SPECIFICATIONS SUMMARY

Table 3-2
Header Specifications

Column	Entry	Explanation
6	H	Header Specification
7-12	SORTR SORTT	Record Sort Tag Sort
13-17	1 - 16383	Decimal number specifies total length of all key fields listed in Field Specifications (must be the maximum for SORTR)
18	A or Blank D	Processing sequence: Ascending or Descending
26	Blank E X	ASCII collating sequence EBCDIC collating sequence ASCII with modification listed in following lines
28	Blank	The key fields will reside at the beginning of each output record
	X	The key fields will be stripped from the beginning of the output records
29-32	Decimal Number	This entry specifies the number of bytes of all key and data fields listed in the Field Specifications (must be maximum for SORTR)

Table 3-3
Record Type Specifications

Column	Entry	Explanation
6	I	The records described in this record specification will be included in the sorting process. If there are no entries in the following columns, all records not yet described will be included
	O	The records described in this record specification will be omitted from the sorting process
7	A	The condition identified by this record specification is a logical AND with the condition from the previous line
	O	The condition identified by this record specification is a logical OR with the condition from the previous line
8	B	Binary - the field contains 2's complement binary data
	C	Character - the field in columns 9-16 (Factor 1) is alphameric
	D	Digit - use the digit value or convert the number to binary
	F	Floating - treat the field as a 2- or 4-word floating binary representation
	I	Same as D but with sign leading and separate, i.e., the first byte of the field is a + or a -
	J	Same as D but with sign trailing and separate, i.e., the last byte of the field is + or -
	K	Same as D but with sign leading overpunch, i.e., sign is superimposed upon first byte of the field
	P	Packed - the field contains packed decimal data
	Z	Zone - use the zone value for the entry

(continued on next page)

Table 3-3 (Cont.)
Record Type Specifications

Column	Entry	Explanation
9-12	Decimal Number	Location of the first byte of the field (Factor 1)
	Blanks	Single byte field
13-16	Decimal Number	Location of last byte of field or location of single byte field
17-18	EQ	Factor 1 must equal Factor 2
	NE	Factor 1 must not equal Factor 2
	LT	Factor 1 must be less than Factor 2
	GT	Factor 1 must be greater than Factor 2
	LE	Factor 1 must be less than or equal to Factor 2
	GE	Factor 1 must be greater than or equal to Factor 2
19	F	The following entry describes a field to be compared with the field specified in columns 9 through 16
	C	The following entry describes a constant to be compared with the field specified in columns 9 through 16
20-23	Decimal Number	Location of the first byte of the field (Factor 2) compared with Factor 1
	Blanks	Factor 2 is a single byte field
24-27	Decimal Number	Position of last byte of Factor 2 field or location of single-byte field
20-39	Character String	The constant compared with Factor 1 entry must agree with mode and length of Factor 1

Table 3-4
Field Specifications

Column	Entry	Explanation
6	F	Field Specification
7	N	Normal - key field sequenced as indicated in column 18 of header
	O	Opposite - key field sequenced opposite to column 18 of header
	F	Forced - single byte key field with special sequence
	D	Data field
8	B	Binary - the key field is in 2's complement binary notation
	C	Character - alphanumeric key or data field
	D	Digit - use digit value or convert to binary for FORTRAN IV numbers
	F	Floating - the key field is in 2- or 4-word floating binary representation
	I	Same as D but with sign leading and separate, i.e., the first byte of the field is a + or a -
	J	Same as D but with sign trailing and separate, i.e., the last byte of the field is + or a -
	K	Same as D but with sign leading over-punch, i.e., sign is superimposed upon first byte of the field
	P	Packed - the key field is in packed decimal
	Z	Zone - zone value
9-12	Decimal Number	Location of first byte of field
18	Replacement Character	The character which will be substituted when the force condition (if any) is met
19	Continuation Character (any character other than blank)	This specification line covers the same forced key field as the preceding line(s)
	Blank	This line designates a new forced key

CHAPTER 4

ERROR CONDITIONS

4.1 ERRORS IN SORT: GENERAL COMMENTS

When SORT exits, it always prints the status of the sort operation. The status message is either SUCCESS or an error message for one of the following errors:

- Command decoder errors
- Specification file errors
- I/O errors

SORT errors are reported at the terminal only and are not recoverable. The format of a SORT error is as follows:

```
SRT -- control-phase:?message   [-RMS-status-code]
```

where:

control-phase	is the SORT phase in control at the time the error occurred. Legal values are: C - command decoder M - merge P - presort
message	is a one-line brief explanation of what happened.
RMS-status-code	is a decimal status code returned by RMS for additional information on file errors only. If RMS is not impacted by the SORT error, this status code does not appear. Status codes likely to be seen are listed with their meanings in Section 4.7.

4.2 COMMAND DECODER ERRORS

If your error message has a control-phase of C, you have made a command decoder error. The command decoder error messages and their probable causes are listed below:

1. SORT COMMAND ERROR

    a. Too many input files (more than two, including specification file) or output files (more than one)

b. General syntax error

c. Too many switches

d. Erroneous switches on the specification file

e. An undefined switch

2. IMPROPER SWITCH: /FI

a. Less than three or greater than eight scratch files.

b. Invalid terminator

NOTE

Valid terminators are period, comma, slash, equal sign and <RET>. "INVALID TERMINATOR" means that some other character was used as a terminator or that SORT expected to find a terminator where none existed.

3. IMPROPER SWITCH: /KE

a. Invalid letter or value

b. Start location or size is 0

c. No period (.) between start location and size

d. Illegal size for data mode

e. Invalid terminator (See NOTE above)

4. TOO MANY KEYS

Buffer space overflowed

NOTE

SORT reserves a buffer area for storage of a table based on the input specifications in order to control the processing of each record. This space should be ample for all situations to make this error unlikely.

5. NO KEYS SPECIFIED

There are no key switches in the command string and no specification file has been declared.

6. KEY AFTER LAST BYTE OF RECORD

The end of an input record key field goes past the stated record size (switch or specification).

7. NO /FO SWITCH

   You omitted the /FORMAT switch on the input file.

8. IMPROPER SWITCH: /FO

   You have not specified a valid format type.

9. IMPROPER SWITCH: /PR

   You specified an invalid sort process.

4.3 SPECIFICATION FILE ERRORS

In general, the detection of an error in the Specification File results in an appropriate message followed by:

- a printout of the entire record, or
- a printout of the record up to the point where the error was detected.

1. INVALID CHARACTER

   a. Column 6 is not H,I,O,F and record is not ALTSEQ.

   b. Process is not SORTR, SORTT, or blank.

   c. Collating sequence is not blank, E, or X.

   d. Data type is not B, C, D, F, I, J, K, P, Z.

   e. Key type is not D, F, N, O.

   f. Logical entry is not A, O, blank, or *.

2. ILLEGAL FIELD

   a. A numeric field in specification contains other than decimal digits or blanks.

   b. No key size is given in Header specification.

   c. No output size is given in Header Specification.

   d. ALTSEQ is misspelled.

   e. ALTSEQ entries do not represent 7-bit octal values.

   f. Last location is less than first location in record field identification.

   g. Size is invalid for data mode.

   h. Sizes of Factors 1 and 2 in Record Specification do not match.

i. Compare relation is undefined.

j. Forced field is other than type C or more than one position.

3. ILLEGAL CONSTANT

a. Constant given in Factor 2 is greater than 20 characters.

b. Mode of constant does not agree with mode of Factor.

c. Invalid characters appear in constant (e.g., non-digits if the constant is numeric).

d. Sign is omitted from binary or packed constant.

4. NO HEADER

a. First record of specification file is not an H specification.

5. INCORRECT SEQUENCE

a. Numeric record sequence is lower than sequence previously encountered.

b. No valid data specification appears when keys are to be stripped from output.

c. Record specification after "include-all" ("include-all" should be last).

d. Key specifications appear after data specifications.

6. NO ALTSEQ

a. Specification for alternate collation entered in Header column 26 but no ALTSEQ Specifications follow.

7. TOO MANY SPECIFICATIONS

a. Number of specifications for a particular type of record have overflowed the buffer space.

NOTE

SORT reserves a buffer space for storage of a table based on the input specifications and used to control the processing of each record type. This space should be ample for all situations to make the error unlikely other than in very exceptional circumstances.

8. LAST RECORD SPEC NOT "I"

   a. Last record specification in file was OMIT.

9. ILLEGAL KEY

   a. A forced field specification was included in an Index SORT (SORTI).

10. NO KEYS SPECIFIED

    a. No key specification appeared before data specifications in a specification file set.

4.4 I/O ERRORS

Once the sort is under way the only normal errors that can occur are due to I/O failure, as indicated below:

1. OPEN (IN/OUT) FAILURE ON filename.ext xxxx

   The file could not be opened by RMS. The parameter xxxx is the appropriate RMS error code.

2. INPUT/OUTPUT ERROR ON filename.ext xxxx

   RMS generated an error while reading or writing this file. The parameter xxxx is the appropriate RMS error code.

3. BAD RECORD SIZE ON filename.ext

   RMS tried to read a record on this file that is larger than the size specified in the /FO switch.

4. NO ROOM IN filename.ext

   a. The storage capacity of the device has been exhausted while writing the output file.

5. INAPPROPRIATE FILE ORGANIZATION

   a. /IN switch was not specified.

6. TEMP FILE ERROR xxxx

   a. An I/O error occurred on a scratch file. xxxx is the RMS error code.

4.5 CONTROL PROGRAM ERRORS

4.5.1 Pre-Sort Errors

1. NO DATA IN INPUT FILE filename.ext

    a. There are no records in the input file.

    b. The file is not an RMS file.

2. INVALID KEY FIELD DATA filename.ext

    a. An inappropriate byte has been found in a field described as D, I, J, K or Z mode.

4.5.2 Merge Errors

1. # OF OUTPUT RECORDS DOES NOT MATCH # INPUT

    a. The number of records released to the sort does not equal the number returned.

NOTE

This is a warning message only. The output file contains as many records as specified in the end-of-sort message.

4.6 OTHER ERRORS

Occasionally, errors may occur due to the rejection of data passed to the SORTS subroutine package. They are reported as:

SORT ERROR -- CODE nn

where nn is the code number returned by a SORTS subroutine (described in paragraph 4.8).

4.7 RMS ERROR CODES

A complete listing of RMS status codes can be found in Appendix D. The status codes that appear in this section represent errors that you will be most likely to encounter.

The status codes in Table 4-1 are due to file primitive failure. Check to see if your file is readable and in the correct format. If these status codes recur, please send an SPR to DIGITAL.

Table 4-1
RMS Status Codes for TRAX

Code	Brief Explanation
-160	Read error on file header attributes
-176	Write error on file header attributes
-400	Deaccess error during $CLOSE
-520	Device positioning error
-560	Enter function failure
-624	File extension failure
-1088	File could not be marked for deletion
-1456	Remove function failure
-1520	Error while reading prologue
-1776	File write error
-1792	Error while writing prologue

The status codes listed in Table 4-2 appear because the file attributes you gave did not match those of the existing file.

Table 4-2
RMS File Attribute Mismatch Status Codes

Code	Brief Explanation
-192	Bucketsize exceeds maximum
-480	Dynamic memory exhausted
-976	Key size equals zero or is too large (indexed file) or is not 4 (relative file)
-1168	Not enough room to open an indexed file

Table 4-3 gives commonly occurring status codes other than RMS.

Table 4-3
Common Non-RMS Status Codes

Code	Brief Explanation
-112	Variable length records on ANSI-labelled magtape are not ANSI D format
-432	RMS tried to access a deleted record
-448	Syntax error, no such device, or device inappropriate for operation
-464	Syntax error in directory name
-496	Directory not found
-512	Device not ready
-672	RMS attempted to create an already existing file
-704	File locked by another user - access denied
-736	File not found
-752	Syntax error in file name
-768	Invalid file options (also occurs if you try to extend a contiguous file)
-784	Device full: cannot create or extend file
-880	Illegal operation - see Appendix D for examples
-896	Illegal record in sequential file or invalid count field
-1008	Magtape is not ANSI-labelled
-1024	Logical channel busy
-1040	Invalid logical channel number
-1120	Maximum record size equals zero during $CREATE
-1152	Not at end of file
-1408	Invalid record address
-1424	Invalid or illegal record format
-1440	Target bucket locked by another task or stream
-1536	Invalid record in indexed file - file may be damaged
-1568	Record size error during $PUT or $UPDATE
-1744	Syntax error in file version number
-1784	Device write-locked

See Appendix D for a more complete explanation of each status code.

4.8 SORTS ERRORS

Whenever SORTS detects an error, it returns an octal non-zero code in the location specified by <Location of Error Code> in the most recent call. In addition, for I/O errors, R1 contains the address of file control block for the file in error. The status supplied by RMS is set into the error status word of that file control block. (See Appendix D for error code values.)

The error codes (octal) and their meanings are listed below:

Error Code	Meaning
00	No errors.
01	Device input error.
02	Device output error.
03	OPEN(IN) failure.
04	OPEN(OUT) failure.
05	Size of current record is greater than maximum size.
06	Not enough work area.
07	RETRN was called after it had exited with a negative error code (end of sort).
10	SORT routine called out of order (The order of the calls should be RSORT, RELES, MERGE, RETRN, ENDS).
11	Sort already in progress (To do a second sort, ENDS must be called to clean up the first sort).
12	Key size is not positive, SORTS detected a zero or negative key size in its calling parameter.
13	Record size not positive.
14	Key address is not even (the keys must start at an even address because SORT uses word moves).
15	Record address is not even.
16	Scratch records will be too large (the size of the keys plus the size of the largest record must be less than 37776 octal).
17	Too few scratch files are given (a minimum of 3 scratch files must be specified).
20	Too many scratch files are given (a maximum of 10 scratch files may be specified).
21	End-of-string record was detected where none was expected.
22	Unexpected end-of-file.
23	SORT found a record larger than expected.
24	Record length is not standard for SORTT.

CHAPTER 5

FILES

5.1 FILE NAME CONVENTIONS

Input, output, and specification files may have any name acceptable to TRAX.

SORT generates its scratch files by using the temporary file creation mechanisms of TRAX. Scratch files are automatically deleted from your file directory by the operating system when SORT is no longer running.

5.2 FILE MAINTENANCE

At the end of a successful run, SORT stores the new output file on the device you specified. The new output file may then be used as immediate input to some other program. The input and specification files are unchanged.

5.3 FILE ALLOCATION

To optimize your sort for speed, SORT tries to allocate in advance the output file and each scratch file at maximum size. SORT does this by making an educated guess about the space needed. Under the tag and record sorting processes, SORT allocates as much space to the output file as to the input file. Each scratch file is arbitrarily given 50 percent more space than the input file, no matter what process is used.

Usually, this pre-allocation process causes no problems. But you should be aware of the instances in which problems due to over-allocation do arise and what you can do about them.

Your pre-allocation size may be too large if one or more of the following conditions are true:

1. You are sorting from an indexed file.

2. You are using OMIT lines in the specification file.

3. You are using the reformatting option in the specification file.

4. You are using a smaller record size in the output file than in the input file.

Some wasting of space in the output file can usually be tolerated, but on a crowded disk you may be notified of insufficient space when there

really was enough to hold your file. Try cutting down on the amount of space by making a conservative estimate of what you need and including that estimate in the /AL switch on the output file side of the command string.

You may perhaps be more concerned about the situation in which you get the following error message when you try to sort a large file:

```
P:?TEMP FILE ERROR -784
```

This means that there is no more room for the allocation of scratch files. Here, too, there may be space wasted due to over-allocation. Try cutting down the size of the scratch file to one third that of the input file. If the error persists, trim the size again. If this does not work, then the disk may be too full to handle the sort.

5.4 SCRATCH FILES

5.4.1 Scratch File Structure

The scratch files consist of one or more sequenced strings of records, followed by an end-of-file flag word. Each string consists of zero or more records, in order by key, followed by an end-of-string flag word. The number of strings on each file varies from time to time as the sort progresses, until, at the end of the merge phase, there is one string on each file. If there is not enough data in the input file to produce at least one string on each scratch file, the unused files will contain only an end-of-file flag word.

The format of the records stored in these files is as follows:

Word 1	Size of the data portion in bytes. The maximum acceptable to SORT is 16K (up to 37777 octal).
Words 2 through n	After possible conversion to a form more suitable for logical comparison, the key fields are stored in reverse.
Words n+1* through m	The data portion for record sort (SORTR) is the record as read from the input file; for SORTT, it is a 2-word relative record pointer.

Two unique records also appear in the scratch files as markers. Each is one word only containing all 0 bits except as below:

- Bit 15 = 1 indicates end-of-string flag
- Bit 14 = 1 indicates end-of-file flag

* When controlled by specification file, word n+1 is used to hold an internal control pointer, and the data starts at n+2.

5.4.2 Using Scratch Files

The scratch files switch, FILES (see Section 3.1.6), is only effective when operating under special conditions. For each scratch file potentially to be used, a certain amount of core is required to establish internal control blocks. This necessarily makes the sort take longer by:

1. Restricting the space available for record storage during the sort, and

2. Forcing the usage of scratch files even for small input files

By specifying a smaller number of scratch files than the default, you may be able to release sufficient space to do the whole sort in core and thereby speed up the sort of a small file.

APPENDIX A

CHARACTER SETS USED BY SORT

Table A-1: The ASCII Character Set with Corresponding EBCDIC Codes

Table A-2: The Subset of the EBCDIC Character Set Used by SORT When EBCDIC Sorting is Requested

Table A-1
The ASCII Character Set with Corresponding EBCDIC Codes

Character	ASCII Code	EBCDIC Code	Character	ASCII Code	EBCDIC Code
NUL	000	000	@	100	174
SOH	001	001	A	101	301
STX	002	002	B	102	302
ETX	003	003	C	103	303
EOT	004	067	D	104	304
ENQ	005	055	E	105	305
ACK	006	056	F	106	306
BEL	007	057	G	107	307
BS	010	026	H	110	310
HT	011	005	I	111	311
LF	012	045	J	112	321
VT	013	013	K	113	322
FF	014	014	L	114	323
CR	015	015	M	115	324
SO	016	016	N	116	325
SI	017	017	O	117	326
DLE	020	020	P	120	327
DC1	021	021	Q	121	330
DC2	022	022	R	122	331
DC3	023	023	S	123	342
DC4	024	074	T	124	343
NAK	025	075	U	125	344
SYN	026	062	V	126	345
ETB	027	046	W	127	346
CAN	030	030	X	130	347
EM	031	031	Y	131	350
SUB	032	077	Z	132	351
ESC	033	047	[	133	112
FS	034	034	\	134	340
GS	035	035	]	135	132
RS	036	036	^	136	137
US	037	037		137	155
SPC	040	100	\	140	171
!	041	117	a	141	201
"	042	177	b	142	202
#	043	173	c	143	203
$	044	133	d	144	204
%	045	154	e	145	205
&	046	120	f	146	206
'	047	175	g	147	207
(	050	115	h	150	210
)	051	135	i	151	211
*	052	134	j	152	221
+	053	116	k	153	222
,	054	153	l	154	223
-	055	140	m	155	224
.	056	113	n	156	225
/	057	141	o	157	226
0	060	360	p	160	227
1	061	361	q	161	230
2	062	362	r	162	231
3	063	363	s	163	242
4	064	364	t	164	243
5	065	365	u	165	244
6	066	366	v	166	245
7	067	367	w	167	246
8	070	370	x	170	247
9	071	371	y	171	250
:	072	172	z	172	251
;	073	136		173	300
<	074	114		174	152
=	075	176		175	320
>	076	156		176	241
?	077	157	DEL	177	007

Table A-2
The Subset of the EBCDIC Character Set Used by SORT when EBCDIC Sorting is Requested[1]

Character	EBCDIC Code	ASCII Code	Character	EBCDIC Code	ASCII Code
NUL	000	000	c	203	143
SOH	001	001	d	204	144
STX	002	002	e	205	145
ETX	003	003	f	206	146
HT	005	011	g	207	147
DEL	007	177	h	210	150
VT	013	013	i	211	151
FF	014	014	j	221	152
CR	015	015	k	222	153
SO	016	016	l	223	154
SI	017	017	m	224	155
DLE	020	020	n	225	156
DC1	021	021	o	226	157
DC2	022	022	p	227	160
DC3	023	023	q	230	161
BS	026	010	r	231	162
CAN	030	030		241	176
EM	031	031	s	242	163
FS	034	034	t	243	164
GS	035	035	u	244	165
RS	036	036	v	245	166
US	037	037	w	246	167
LF	045	012	x	247	170
ETB	046	027	y	250	171
ESC	047	033	z	251	172
ENQ	055	005		300	173
ACK	056	006	A	301	101
BEL	057	007	B	302	102
SYN	062	026	C	303	103
EOT	067	004	D	304	104
DC4	074	024	E	305	105
NAK	075	025	F	306	106
SUB	077	032	G	307	107
SPC	100	040	H	310	110
[	112	133	I	311	111
.	113	056		320	175
<	114	074	J	321	112
(	115	050	K	322	113
+	116	053	L	323	114
!	117	041	M	324	115
&	120	046	N	325	116
]	132	135	O	326	117
$	133	044	P	327	120
*	134	052	Q	330	121
)	135	051	R	331	122
;	136	073	\	340	134
↑ or ^	137	136	S	342	123
-	140	055	T	343	124
/	141	057	U	344	125
	152	174	V	345	126
,	153	054	W	346	127
%	154	045	X	347	130
or	155	137	Y	350	131
>	156	076	Z	351	132
?	157	077	0	360	060
	171	140	1	361	061
:	172	072	2	362	062
#	173	043	3	363	063
@	174	100	4	364	064
'	175	047	5	365	065
=	176	075	6	366	066
"	177	042	7	367	067
a	201	141	8	370	070
b	202	142	9	371	071

[1] This table shows only those EBCDIC characters for which there are ASCII equivalents. It therefore illustrates the collating sequence SORT applies when the Header Specification contains E in column 26.

APPENDIX B

PRINTABLE CHARACTERS

Table B-1: Printable Characters Grouped by Equal Digits

Table B-2: Printable Characters Grouped by Equal Zones

Table B-1
Printable Characters Grouped by Equal Digits

Group	Character	DEC 029 Card Code	Group	Character	DEC 029 Card Code
0	blank	No punches	5	E	12-5
	&	12		N	11-5
	-	11		V	0-5
	0	0		5	5
1	A	12-1	6	F	12-6
	J	11-1		O	11-6
	/	0-1		W	0-6
	1	1		6	6
2	B	12-2	7	G	12-7
	K	11-2		P	11-7
	S	0-2		X	0-7
	2	2		7	7
3	C	12-3	8	H	12-8
	L	11-3		Q	11-8
	T	0-3		Y	0-8
	3	3		8	8
4	D	12-4	9	I	12-9
	M	11-4		R	11-9
	U	0-4		Z	0-9
	4	4		9	9

Table B-2
Printable Characters Grouped by Equal Zones

Group	Character	DEC 029 Card Code	Group	Character	DEC 029 Card Code
0	&	12	2	0	0
	(	12-5-8		%	0-4-8
	+	12-6-8		'	0-3-8
	.	12-3-8		/	0-1
	<	12-4-8		>	0-6-8
	l	12-7-8		?	0-7-8
	A	12-1		S	0-2
	B	12-2		T	0-3
	C	12-3		U	0-4
	D	12-4		V	0-5
	E	12-5		W	0-6
	F	12-6		X	0-7
	G	12-7		Y	0-8
	H	12-8		Z	0-9
	I	12-9			
1	-	11	3	Blank	No punches
	!	11-2-8		"	7-8
	$	11-3-8		#	3-8
	)	11-5-8		'	5-8
	*	11-4-8		:	2-8
	;	11-6-8		=	6-8
	J	11-1		@	4-8
	K	11-2		1	1
	L	11-3		2	2
	M	11-4		3	3
	N	11-5		4	4
	O	11-6		5	5
	P	11-7		6	6
	Q	11-8		7	7
	R	11-9		8	8
				9	9

APPENDIX C

SORT PROGRAM EXAMPLES

EXAMPLE 1: SALES LIST

Input Record Positions

1-5	Customer Number (CUSTNO)
6-39	Customer Name (CUSTNM)
52-59	Date
60-65	Item Number (ITEMNO)
66-70	Item Quantity (ITEMQY)
71-77	Price
78-80	Blanks

This Record Sort is primarily intended to produce reformatted records. The structure of the input file is indicated above.

The input file contains only one record type, therefore no Record Type Specifications are needed. The keys for the sort are CUSTNO, ITEMNO, and ITEMQY, with the last key sorted in descending sequence. The output file will have records ordered from lowest to highest customer number, with the lowest to highest item number records within each customer number category, and the highest to lowest item quantity in each item number category.

The order of the data field specifications reformat the output record. Note the use of the blank in columns 78 and 79; the data will be set off in field columns.

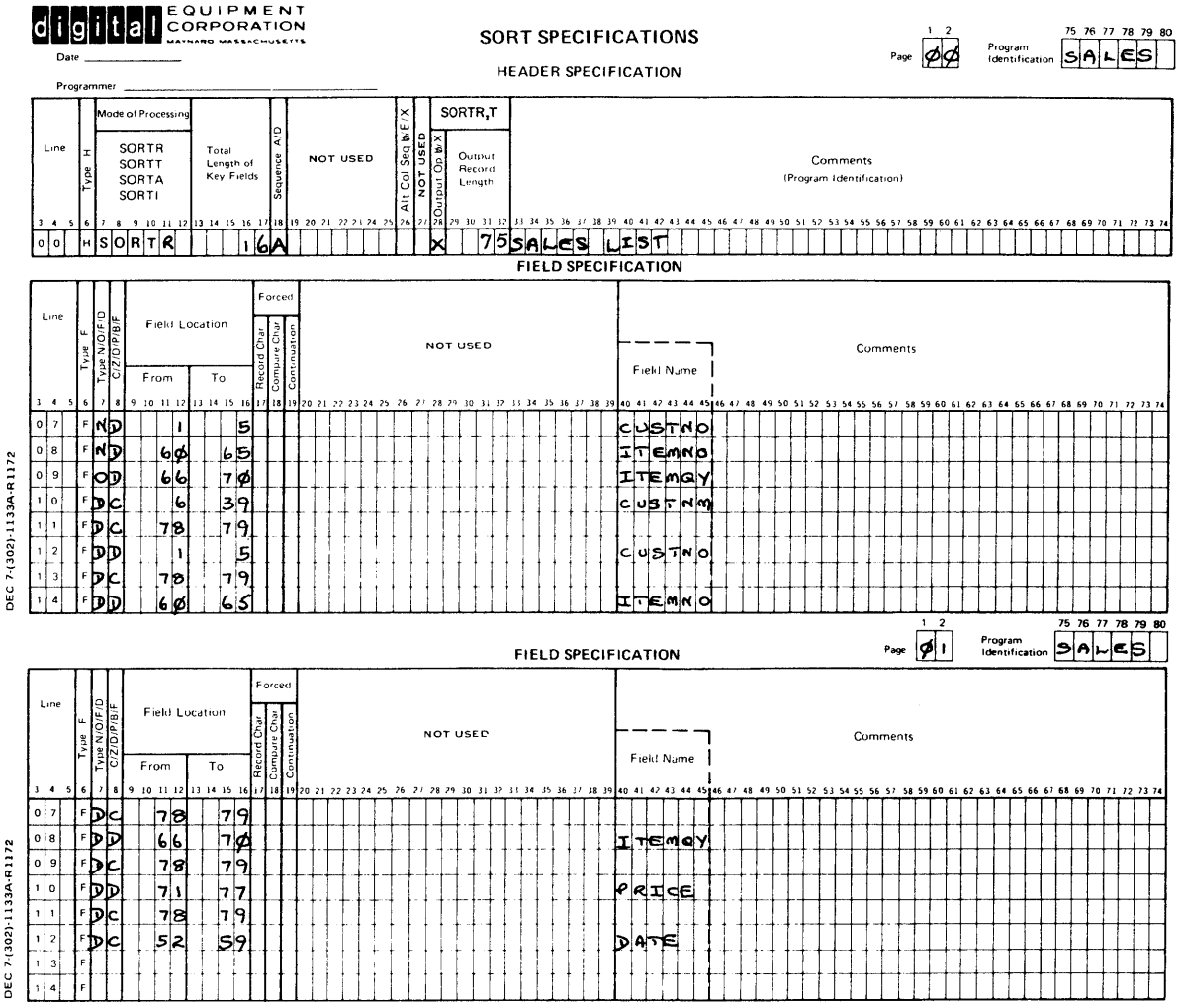


Figure C-1 Sales List

EXAMPLE 2: OVERTIME ANALYSIS

Input Record Positions

1-3	Department Number (DEPTNO)
4-8	Employee Number (EMPNUM)
9-39	Employee Name (EMPNAM)
40-43	Standard Number of Working Hours (STDHRS)
50-53	Total Number of Hours Worked (HOURS)
80	Plant Identification Code

This Record Sort produces a record of employees by department number in the main plant (M) who incurred overtime. The Record Type Specification shows the use of both an Include and an Omit line; with a field and a constant Factor 2 comparison.

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

SORT SPECIFICATIONS

Date _____

Programmer: _____

Page 00 Program Identification OTANAL

HEADER SPECIFICATION

Line	Type H	Mode of Processing: SORTR / SORTT / SORTA / SORTI	Total Length of Key Fields	Sequence A/D	NOT USED	Alt Col Seq W/E/X	NOT USED	Output Op W/X	SORTR,T Output Record Length	Comments (Program Identification)
00	H	SORTR	12A						470	OVERTIME ANALYSIS (MAIN PLANT)

RECORD TYPE SPECIFICATION

Line	Type I/O	Continuation A/O	C/Z/D/P/B/F	Factor 1 Field Location From	Factor 1 Field Location To	EQ NE GT GE LT LE	F/C	Factor 2 Constant / Field Location From	Factor 2 Field Location To	Record Name	Comments
01	O		D	50	53	LEF		40	43	HOURS	MUST EXCEED STANDARD
02	H		C		80	EQ	CM			LOCATION	MAIN PLANT ONLY
03											
04											
05											
06											

FIELD SPECIFICATION

Line	Type F	Type N/O/F/D	C/Z/D/P/B/F	Field Location From	Field Location To	Forced Record Char	Forced Compare Char	Forced Continuation	NOT USED	Field Name	Comments
07	F	N	D	1	3					DEPTNO	
08	F	O	D	50	53					HOURS	
09	F	N	D	4	8					EMPNUM	
10	F	D	C	9	39					EMPNAM	
11	F	D	D	40	43					STDHRS	
12	F										
13	F										
14	F										

DEC 7-(302)-1133A-R1172

Figure C-2 Overtime Analysis

EXAMPLE 3: EMPLOYEE LIST

This Tag Sort illustrates the use of forced key field specifications. The sort is based on the plant identification code in column 80 of the input records. In order to sort them in the order M, W, P, the M in column 80 is forced to a 1, the W is forced to a 2, and the P is forced to a 3. The output record entry is unchanged.

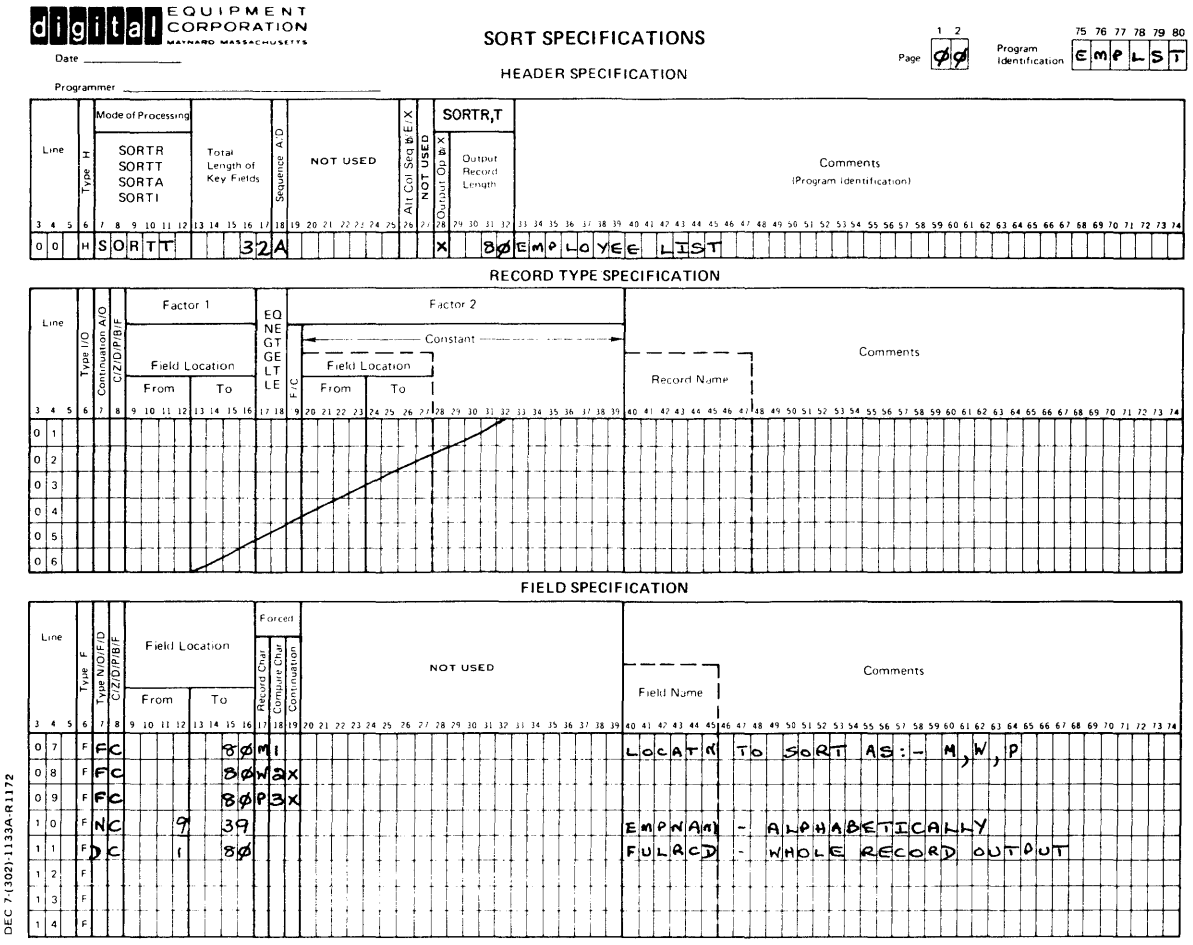


Figure C-3 Employee List

APPENDIX D

RMS I/O STATUS CODES

This appendix describes I/O status codes that can be returned by RMS.

For some error conditions, RMS uses the RMS-status-code field of the SORT error message to communicate additional information. Table D-1 lists the RMS status codes and what they mean.

Table D-1
RMS Error Status Codes

Decimal Value	Description
-16	Operation aborted: out-of-stack save area or in-core data structures corrupted.
-32	ACP could not access the file.
-48	File activity precludes action (e.g., attempting to close a file with outstanding asynchronous record operation).
-64	Bad area identification number (AID) field in allocation XAB (i.e., out of sequence).
-80	Illegal value in alignment boundary type (ALN) field of allocation XAB.
-96	Value in allocation quantity (ALQ) field in FAB (or allocation XAB) exceeds maximum or, during an explicit $EXTEND operation, equals 0.
-112	Records in a file on ANSI-labeled magnetic tape are variable length but not in ANSI D format.
-128	Illegal value in allocation options (AOP) field in allocation XAB.
-144	Invalid operation at AST level: attempting to issue a synchronous operation from an asynchronous record operation completion routine.
-160	Read error on file header attributes.
-176	Write error on file header attributes.

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-192	Bucket size (BKS) field in FAB contains value exceeding maximum.
-208	Bucket size (BKZ) field in allocation XAB contains value exceeding maximum.
-224	Block length (BLN) field in a FAB or RAB is incorrect.
-232	Beginning of file detected on $SPACE operation to magnetic tape file.
-240	Private buffer pool address not a double word boundary.
-256	Private buffer pool size not a multiple of 4.
-272	Internal error detected in RMS-11; no recovery possible; contact a Software Specialist.
-288	Cannot connect RAB (i.e., only one record access stream permitted for sequential files).
-304	$UPDATE attempting to change a key field that does not have the change attribute.
-320	Index file bucket check-byte mismatch. The bucket has been corrupted. No recovery possible for the bucket.
-352	Invalid COD field in XAB or this XAB type is illegal for the organization or operation.
-368	ACP could not create file.
-384	No current record: operation not immediately preceded by a successful $GET or $FIND.
-400	ACP deaccess error during $CLOSE.
-416	Invalid area number in DAN field of key definition XAB.
-448	1. Syntax error in device name. 2. No such device. 3. Inappropriate device for operation (e.g., attempting to create an indexed file on magnetic tape).
-464	Syntax error in directory name.
-480	Dynamic memory exhausted: insufficient space in central space pool or private buffer pool.

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-496	Directory not found.
-512	Device not ready.
-520	Device positioning error.
-544	Duplicate key detected, duplicates allowed attribute not set for one or more key fields.
-560	ACP enter function failed.
-576	Environment error; operation not selected in ORG$ macro.
-592	End of file.
-608	Expanded string area in NAM block too short.
-616	File expiration date not reached.
-624	File extend failure.
-640	Not a valid FAB: BID field does not contain FB$BID.
-656	1. Record operation attempted was not declared in FAC field of FAB at open time. 2. Invalid contents in FAC field. 3. FB$PUT not present in FAC for $CREATE operation.
-672	File already exists (attempted $CREATE operation).
-680	Invalid file ID.
-688	Invalid combination of values in FLG field of key definition XAB (e.g., no duplicates and keys can change).
-704	File locked by another user -- your program cannot access the file because its sharing specification cannot be met.
-720	ACP Find function failed.
-736	File not found.
-752	Syntax error in file name.
-768	Invalid file options.
-784	Device full: cannot create or extend file.
-800	Invalid area number in IAN field of key definition XAB.
-816	Index not initialized (this code can only occur in the STV field when STS contains ER$RNF).

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-832	Invalid IFI field in FAB.
-848	Maximum number (254) of key definition or allocation XABs exceeded or multiple summary, protection, or date XABs present during operation.
-864	$INIT or $INITIF macro call never issued.
-880	Illegal operation; examples include: 1. Attempting a $TRUNCATE operation to a non-sequential file. 2. Attempting an $ERASE or $EXTEND operation to a magnetic tape file. 3. Issuing a block mode operation (e.g., $READ or $WRITE) to a stream not connected for block operations. 4. Issuing a record operation (e.g., $GET, $PUT) to a stream connected for block mode operations.
-896	Illegal record encountered in sequential file: invalid count field.
-912	Invalid internal stream identifier (ISI) field in RAB (field may have been altered by user) or $CONNECT never issued for stream.
-928	Key buffer address (KBF) field equals 0.
-944	Record identifier (i.e., the 4-byte location addressed by KBF) for random operation to relative file is 0 or negative.
-960	Invalid key of reference (KRF) in RAB: 1. As input to random $GET or $FIND operation, or 2. As input to $CONNECT or $REWIND (in this case, ER$KRF is returned for the first record operation following the $CONNECT or $REWIND).
-976	Key size equals 0 or too large (indexed file) or not equal to 4 (relative file).
-992	Invalid area number in LAN field of key definition XAB.
-1008	Magnetic tape is not ANSI labeled.
-1024	Logical channel busy.
-1040	Invalid value in logical channel number (LCH) field of FAB.
-1048	Attempt to extend an area containing an unused extent.
-1056	Invalid value in LOC field of allocation XAB.

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-1072	In-core data structures (e.g., I/O buffers) corrupted (this code can only occur in the STV field when STS contains ER$ABO).
-1088	ACP could not mark file for deletion.
-1104	1. Maximum record number field contains a negative value during $CREATE of relative file. 2. Record identifier (pointed to by KBF) for random operation to relative file exceeds maximum record number specified when file created.
-1120	Maximum record size (MRS) field contains 0 during $CREATE operation and: 1. Record Format is fixed, or 2. File organization is relative.
-1136	Odd address in Name Block address (NAM) field in FAB on $OPEN, $CREATE, or $ERASE.
-1152	Not at end-of-file: attempting a $PUT operation to a sequential file when stream is not positioned to EOF.
-1168	Cannot allocate internal index descriptor: insufficient room in space pool while attempting to open an indexed file.
-1184	No primary key definition XAB present during $CREATE of indexed file.
-1216	XABs in chain not in correct order: 1. Allocation or key definition XABs not in ascending (or densely ascending) order. 2. XAB of another type intervenes in key definition or allocation XAB sub-chain.
-1232	Invalid value in file organization (ORG) field of FAB.
-1248	Error in file prologue: file is corrupted and must be reconstructed.
-1264	Key position (POS) field in key definition XAB contains a value exceeding maximum record size.
-1280	File header contains bad date and time information (retrieved by RMS-11 because a date and time XAB is present during a $OPEN or $DISPLAY operation); file may be corrupted.

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-1296	Privilege violation: access to the file denied by the operating system.
-1312	Not a valid RAB: BID field does not contain RB$BID. Refer to Section A.3 of this Appendix.
-1328	1. Illegal values in record access mode (RAC) field of RAB. 2. Illogical value in RAC field (e.g., RB$KEY with a sequential file).
-1344	1. Illegal values in record attributes (RAT) field of FAB during $CREATE. 2. Illogical combination of attributes (e.g., FB$CR and FB$FTN) in RAC field during $CREATE.
-1360	Record address (RBF) field in RAB contains an odd address (block mode access only).
-1376	File read error.
-1392	Record already exists: during a $PUT operation in random mode to a relative file, an existing record found in the target record position.
-1408	Invalid RFA in RFA field of RAB during RFA access.
-1424	1. Invalid record format in RFM field of FAB during $CREATE. 2. Specified record format is illegal for file organization.
-1440	Target bucket locked by another task or another stream in the same program.
-1456	ACP Remove function failed.
-1472	Record identified by KBF/KSZ fields of RAB for random $GET or $FIND operation does not exist in relative or indexed file (for indexed files only, STV may contain ER$IDX). Record may never have been written or may have been deleted.
-1488	$FREE operation issued but no bucket was locked by stream.
-1504	Record options (ROP) field contains illegal values or illogical combination of values.
-1520	Error while reading prologue.
-1536	Invalid PRV record encountered in indexed file; file may be corrupted.
-1552	Record stream active, i.e., in asynchronous environment, attempting to issue a record operation to a stream that has a request outstanding.

(Continued on next page)

Table D-1 (Cont.)
RMS Error Status Codes

Decimal Value	Description
-1568	Record size specified in RSZ of RAB during $PUT or $UPDATE is invalid: 1. RSZ equals 0. 2. RSZ exceeds maximum record size (MRS) specified when file created. 3. RSZ not equal to size of Current Record for $UPDATE operation to a sequential file on disk. 4. RSZ does not equal MRS (for fixed format records).
-1584	Record too big for your buffer: RMS-11 could not move entire record retrieved by $GET operation to work area (UBF/USZ). Note that this error does not destroy the current context of the stream. Rather, the stream's context is updated as if the operation had been completely successful.
-1600	During $PUT operation, key of record to be written is not equal to or greater than key of previous record (and RAC field contains RB$SEQ).
-1616	Illogical value in SHR field of FAB (e.g., FB$WRI specified for sequential file).
-1632	Invalid SIZ field in key definition XAB during $CREATE (e.g., specified size exceeds maximum record size).
-1648	During asynchronous record operation, RMS-11 has found that the stack is too big to be saved (this code can only occur in the STV field when STS contains ER$ABO).
-1664	System directive error.
-1680	Index tree error: indexed file is corrupted.
-1696	Syntax error in file type (e.g., more than three characters specified).
-1712	Invalid address in UBF field of RAB: 1. UBF contains 0, or 2. UBF not word aligned (for block mode access only).
-1728	Invalid USZ field in RAB (i.e., USZ contains 0).
-1744	Syntax error in file version number.
-1760	Invalid VOL field in allocation XAB (i.e., VOL does not contain 0).
-1776	File write error.
-1784	Device is write locked.
-1792	Error while writing prologue.
-1808	XAB field in FAB (or NXT field in XAB) contains an odd address.

GLOSSARY

Alphanumeric Characters - the numbers and letters contained in the ASCII character set.

ASCII Character Set - the set of 128 seven-bit American Standard Code for Information Interchange characters (see Appendix A).

Byte - the smallest addressable unit of information; eight bits. Each ASCII character usually resides in a single byte.

Collating Sequence - the order into which characters are sorted based upon numeric values assigned to each.

Device - any of the peripherals referenced by the operating system.

Digit - in EBCDIC representation, the numeric value of the low-order four bits of a byte.

EBCDIC - an alternate character set used widely in the computing industry (see Table A-2 for comparison with ASCII characters).

Field - a logically distinguishable area within a record in which data is located.

File - a collection of related records treated as a unit.

Floating-Point Binary - a method for storing numeric data, the exponent occupies the first byte and the mantissa resides in the next three or seven bytes.

Format - the arrangement of any record or file; the order in which fields reside in a record.

Key, Key Field - the sorting key is the data element chosen from a record to control the sort. For example, if a list of names is sorted alphabetically, the keys would be each successive letter in the name field.

Key, Major - the most important field in the total key. If you were sorting a list by name, department and salary, name would be the major key.

Key, Minor - the least significant field in the total key. In the example above, salary is the minor key.

Line - In this document, a line generally refers to a line in the SORT specifications form or a record on the specification file.

Packed Decimal - method for compact storage of numeric data; two digits are stored in each 8-bit byte and the sign resides in the last byte of the low-order digit.

RMS - Record Management Services

Record - the unit of information in a file; a group of related fields treated as a unit

System Device - the device on which the Executive resides.

Word - two bytes or sixteen bits of data

Work File - a collection of sorted records created during the processing cycle and released after the sort is finished

Zone - the top three punches of a character as represented on a punch card or the numeric value of the upper four bits of a byte

INDEX

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an 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 _____

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 Documentation
146 Main Street ML5-5/E39
Maynard, Massachusetts 01754