# PDP-11
# SORT Reference Manual

Order No. AA-3341C-TC

digital

May 1977

This document discusses the features and uses of the PDP-11 SORT utility. Installation procedures for the operating systems listed below are included. This document and the software to which it pertains are for use with RMS-11-formatted files only.

# PDP-11
# SORT Reference Manual

Order No. AA-3341C-TC

| | |
|---|---|
| **SUPERSESSION/UPDATE INFORMATION:** | This document supersedes the document of the same name, Order No. DEC-11-USTMA-B-D, published July 1975 for the operating systems listed below only. |
| **OPERATING SYSTEM AND VERSION:** | RSX-11M V03 and V3.1<br>RSTS/E V06B<br>IAS V02 |
| **SOFTWARE VERSION:** | SORT-11 V02 |

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, October 1974
Revised: July 1975
May 1977

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.



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

The following are trademarks of Digital Equipment Corporation:

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

CONTENTS


| | | | Page |
|---|---|---|---|
| PREFACE | | | vii |
| CHAPTFR | 1 | OVERVIEW | 1-1 |
| | 1.1 | INTRODUCTION | 1-1 |
| | 1.1.1 | Data Files | 1-2 |
| | 1.1.2 | Command String and Specification File | 1-4 |
| | 1.2 | HOW SORT OPERATES | 1-7 |
| | 1.3 | SORT PROCESSING OPTIONS | 1-8 |
| | 1.3.1 | Record Sort (SORTR) | 1-8 |
| | 1.3.2 | Tage Sort (SORTT) | 1-9 |
| | 1.3.3 | Address Routing Sort (SORTA) | 1-9 |
| | 1.3.4 | Index Sort (SORTI) | 1-10 |
| | 1.3.5 | Graphic Representation of SORTA and SORTI Output | 1-10 |
| CHAPTER | 2 | THE SORT COMMAND STRING | 2-1 |
| | 2.1 | INTRODUCTION | 2-1 |
| | 2.2 | HOW TO RUN SORT | 2-1 |
| | 2.2.1 | Running SORT in Interactive Mode | 2-1 |
| | 2.2.2 | Running SORT in RSTS/E Batch Mode | 2-2 |
| | 2.3 | THE COMMAND STRING FORMAT | 2-2 |
| | 2.3.1 | File Specifications | 2-3 |
| CHAPTER | 3 | CONTROLLING SORT OPTIONS | 3-1 |
| | 3.1 | SWITCHES | 3-1 |
| | 3.1.1 | ALLOCATION (/AL:n) | 3-1 |
| | 3.1.2 | BLOCKSIZE (/BL:n) | 3-1 |
| | 3.1.3 | BUCKETSIZE (/BU:n) | 3-1 |
| | 3.1.4 | CONTIGUOUS (/CO) | 3-3 |
| | 3.1.5 | DEVICE (/DE:x) | 3-3 |
| | 3.1.6 | FILES (/FI:n) | 3-3 |
| | 3.1.7 | FORMAT (/FO:x:n) | 3-3 |
| | 3.1.8 | INDEXED SEQUENTIAL (/IN:X) | 3-3 |
| | 3.1.9 | KEY (/KE:abm.n) | 3-4 |
| | 3.1.10 | PROCESS (/PR:x) | 3-5 |
| | 3.1.11 | RELATIVE (/RE) | 3-5 |
| | 3.1.12 | SEQUENTIAL (/SE) | 3-5 |
| | 3.1.13 | SIZE (/SI:n) | 3-5 |
| | 3.2 | THE SPECIFICATION FILE | 3-5 |
| | 3.2.1 | Header Specification | 3-9 |
| | 3.2.1.1 | Notes and Comments on Header Specification Entries | 3-10 |
| | 3.2.1.2 | ALTSEQ Format and Notes | 3-10 |
| | 3.2.2 | Record Type Specification | 3-11 |
| | 3.2.2.1 | Notes and Comments on Record Type Specifications | 3-13 |
| | 3.2.3 | Field Specifications | 3-14 |
| | 3.2.3.1 | Notes and Comments on Field Specification Entries | 3-16 |

CONTENTS (Cont.)


| | | | Page |
|---|---|---|---|
| | 3.2.4 | Using a Specification File | 3-17 |
| | 3.3 | SORT SPECIFICATION SUMMARY | 3-19 |
| CHAPTER | 4 | ERROR CONDITIONS | 4-1 |
| | 4.1 | ERRORS IN SORT: GENERAL COMMENTS | 4-1 |
| | 4.2 | COMMAND DECODER ERRORS | 4-1 |
| | 4.3 | SPECIFICATION FILE ERRORS | 4-3 |
| | 4.4 | I/O ERRORS | 4-5 |
| | 4.5 | CONTROL PROGRAM ERRORS | 4-6 |
| | 4.5.1 | Pre-Sort Errors | 4-6 |
| | 4.5.2 | Merge Errors | 4-6 |
| | 4.6 | OTHER ERRORS | 4-6 |
| | 4.7 | RMS ERROR CODES | 4-7 |
| CHAPTER | 5 | INTERNAL OPERATION | 5-1 |
| | 5.1 | FILES | 5-1 |
| | 5.1.1 | User and Scratch File Name Conventions | 5-1 |
| | 5.1.2 | LUN Assignments | 5-1 |
| | 5.1.3 | File Maintenance | 5-2 |
| | 5.1.4 | File Allocation | 5-2 |
| | 5.1.5 | Scratch File Structure | 5-3 |
| | 5.1.6 | Scratch Files Use | 5-3 |
| | 5.2 | ORGANIZATION OF SORT | 5-4 |
| | 5.3 | SORTS ERRORS | 5-4 |
| | 5.4 | USING THE SORTS SUBROUTINE PACKAGE | 5-5 |
| | 5.4.1 | Conventions and Standards | 5-6 |
| | 5.4.1.1 | RSORT - Initializing the Sort | 5-6 |
| | 5.4.1.2 | RELES - Passing Input Records to the Sort | 5-7 |
| | 5.4.1.3 | MERGE - Merging the Scratch Files | 5-8 |
| | 5.4.1.4 | RETRN - Requesting an Output Record | 5-8 |
| | 5.4.1.5 | ENDS - Ending the Sort | 5-9 |
| | 5.4.2 | Setting up the Keys | 5-9 |
| | 5.4.3 | Calculating the Size of the Work Area | 5-10 |
| | 5.4.4 | Typical Calling Sequences | 5-10 |
| | 5.4.5 | Linking SORTS Subroutines with Your Program | 5-10 |
| APPENDIX | A | CHARACTER SETS USED BY SORT | A-1 |
| APPENDIX | B | PRINTABLE CHARACTERS | B-1 |
| APPENDIX | C | SORT PROGRAM EXAMPLES | C-1 |
| APPENDIX | D | INTERNAL OPERATIONS | D-1 |
| | D.1 | PROCESS DESCRIPTION | D-1 |
| | D.2 | THE SORT WORK AREA | D-1 |
| | D.3 | PERFORMANCE PARAMETERS | D-4 |
| APPENDIX | E | SORT INSTALLATION | E-1 |
| | E.1 | GENERAL COMMENTS | E-1 |
| | E.2 | SORT KIT CONTENTS | E-1 |
| | E.2.1 | Building the SORT Task for RSX-11M or IAS | E-2 |
| | E.2.2 | Building the SORT Task for RSTS/E | E-2 |
| | E.3 | CHANGING THE SORT TASK IMAGE SIZE | E-3 |

CONTENTS (Cont.)


| | | Page |
|---|---|---|
| APPENDIX F | RMS I/O STATUS CODES | F-1 |
| GLOSSARY | | Glossary-1 |
| INDEX | | Index-1 |


FIGURES


| | | | |
|---|---|---|---|
| FIGURE | 1-1 | Sample Record Types | 1-3 |
| | 3-1 | Specification File Format | 3-6 |
| | 3-2 | SORT Specification Form | 3-8 |
| | 3-3 | SORT Specification Entries | 3-18 |
| | C-1 | Sales List | C-2 |
| | C-2 | Overtime Analysis | C-3 |
| | C-3 | Employee List | C-4 |
| | C-4 | Bonus Analysis | C-5 |
| | C-5a | Stock Order List, Page 1 | C-6 |
| | C-5b | Stock Order List, Page 2 | C-7 |
| | D-1 | SORT Work Area Allocation | D-3 |


TABLES


| | | | |
|---|---|---|---|
| TABLE | 1-1 | Selecting the Sorting Process and Devices that Best Suit the Processing Environment | 1-3 |
| | 1-2 | Sorted Output File | 1-4 |
| | 1-3 | Sorting Process Options | 1-9 |
| | 2-1 | Device Specification Codes | 2-4 |
| | 3-1 | Switch Summary | 3-2 |
| | 3-2 | Header Specifications | 3-19 |
| | 3-3 | Record Type Specifications | 3-20 |
| | 3-4 | Field Specifications | 3-22 |
| | 4-1 | RMS Status Codes for RSX-11M | 4-7 |
| | 4-2 | RMS File Attribute Mismatch Status Codes | 4-7 |
| | 4-3 | Other Commonly Occurring Status Codes | 4-8 |
| | A-1 | The ASCII Character Set with Corresponding EBCDIC Codes | A-2 |
| | A-2 | The Subset of the EBCDIC Character Set used by SORT when EBCDIC Sorting is Requested | A-3 |
| | B-1 | Printable Characters Grouped by Equal Digits | B-2 |
| | B-2 | Printable Characters Grouped by Equal Zones | B-3 |
| | D-1 | User-Accessible SORT Parameters | D-5 |
| | F-1 | RMS Error Status Codes | F-1 |

## PREFACE

This reference manual describes the features and operation of the SORT utility program. 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 contains more detailed information about the SORT program for special applications.

Appendices A and B contain tables of ASCII and EBCDIC, and zone/digit card punch codes respectively. Appendix C contains several examples of SORT programs. Appendix D discusses internal SORT operations, mainly for the benefit of experienced programmers. Appendix E contains SORT installation information for all systems. Appendix F contains RMS-returned status codes.

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

CHAPTER 1

# OVERVIEW

## 1.1 INTRODUCTION

The SORT utility program allows you to read any input file, to sort the contents, and to write out the sorted data onto an output file. The sorting sequence is determined by control fields, also known as 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 four sorting techniques:

- Record Sort (SORTR) 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.

- Tag Sort (SORTT) 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.

- Address Routing Sort (SORTA) produces an address file without reordering the input file. The address file, sorted by record keys, can later be used as an index file* to read the data base in the desired sequence. Any number of address files may be created for the same data base. A customer master file, for instance, may be referenced by customer-number index or sales-territory index for different reports. SORTA is the fastest of the four sorting methods.

- Index Sort (SORTI) produces an index file* containing the key field of each data record and a pointer to its location in the input file. The index file can be used for sequential or direct accessing from a random file.

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.

---

* Not indexed by RMS-11.

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.

### 1.1.1 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. Card reader
4. Paper tape reader
5. 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<br>(Record Sort) | Disk<br>Magtape<br>Paper Tape<br>Cards<br>Console | Disk<br>Magtape[1]<br>Paper Tape<br>Printer<br>Console | Disk<br>(3-8 files) |
| SORTT<br>(Tag Sort) | Disk<br>Magtape | Disk<br>Magtape[1]<br>Printer<br>Console<br>Paper Tape | Disk<br>(3-8 files) |
| SORTA<br>(Address Routing Sort) | Disk<br>Magtape | Disk | Disk<br>(3-8 files) |
| SORTI<br>(Index Sort) | Disk<br>Magtape | Disk | Disk<br>(3-8 files) |

[1]Provided records are at least 18 bytes long. Magtape must be in ANSI format.

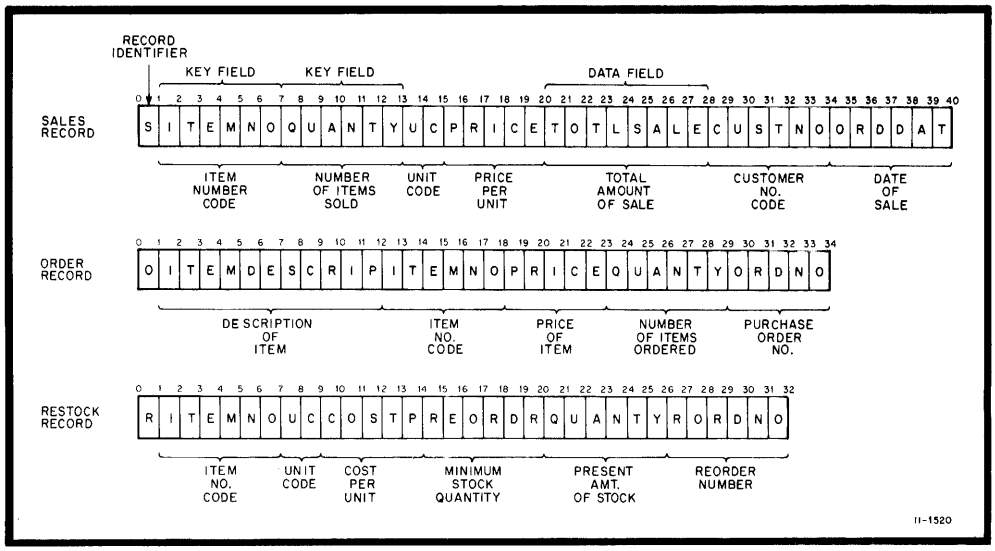



Figure 1-1 Sample Record Types

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 illustrated in Table 1-2. Please note that data fields other than those you use for keys, salesperson's code for instance, may be in any order.

Table 1-2
Sorted Output File

| Major Key: Item Number | Minor Key: Quantity |
|---|---|
| Lowest item no. | Lowest quantity |
| | Next higher quantity |
| . | . |
| . | . |
| . | . |
| . | . |
| Lowest item no. | Highest quantity |
| Next higher item no. | Lowest quantity |
| | Next higher quantity |
| . | . |
| . | . |
| . | . |
| . | . |
| Next higher item no. | Highest quantity |
| Highest item no. | Lowest quantity |
| | Next higher quantity |
| . | . |
| . | . |
| . | . |
| . | . |
| Highest item no. | Highest quantity |

### 1.1.2 Command String and Specification File

You can direct 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

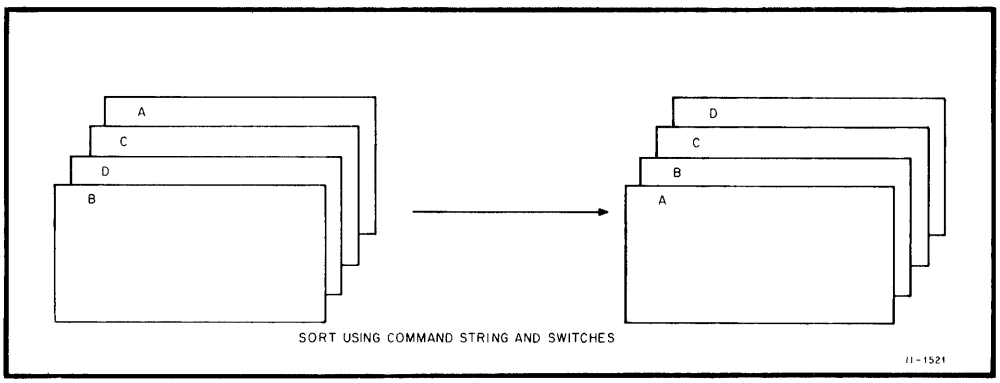
3. To reference a specification file or to specify other switches to control the sort

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. Two additional switches may be used 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 only require a command string and switches.

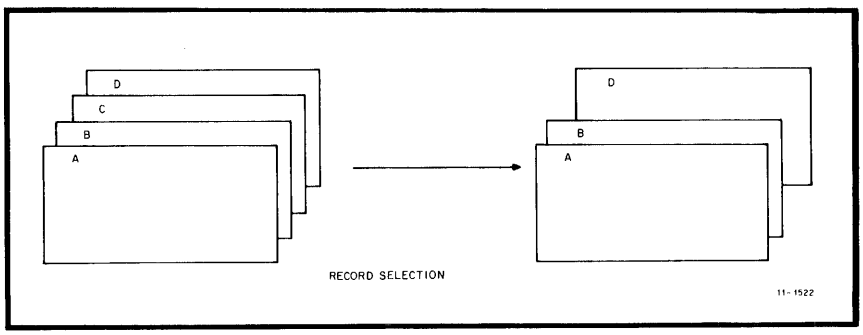


SORT USING COMMAND STRING AND SWITCHES
11-1521

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

The specification file gives you a variety of controlling features. They are listed below:

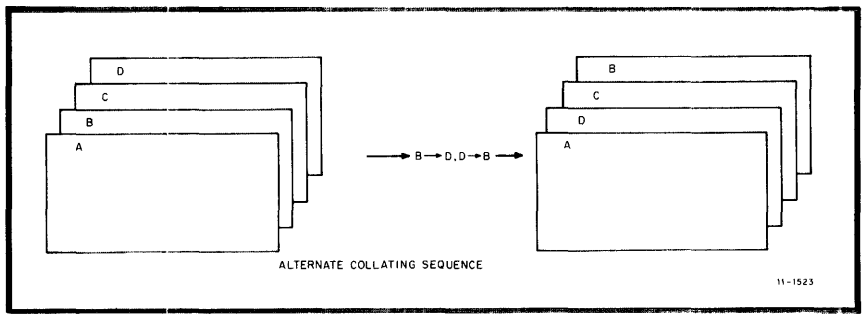
1. Record Selection



RECORD SELECTION
11-1522

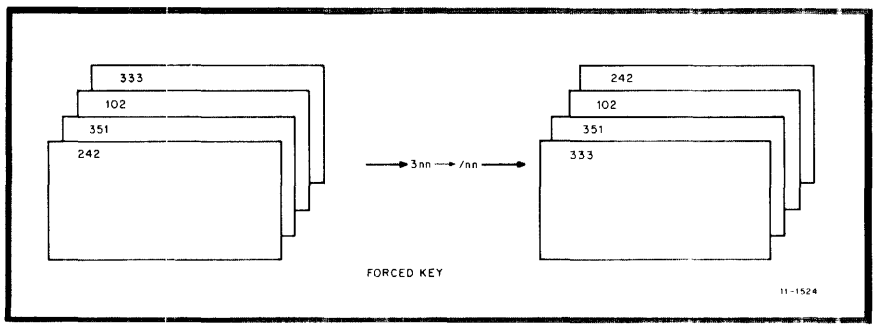
You can include or omit any records from the sorting process. The output file will contain only the records that you specify.

2. Alternate Collating Sequence



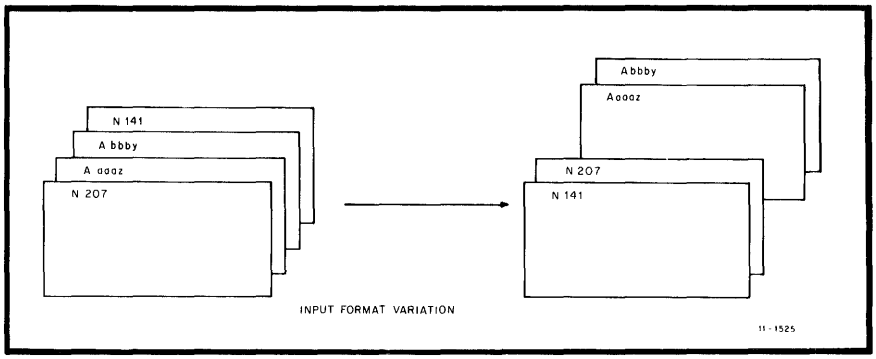

If necessary, you can specify an alternate collating sequence. The normal sequence is that implied in ASCII code. One alternate choice is EBCDIC values. The other is an individual alternate collating sequence (ALTSEQ). An ALTSEQ can be used to change the ASCII values of the normal sequence. It applies 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.

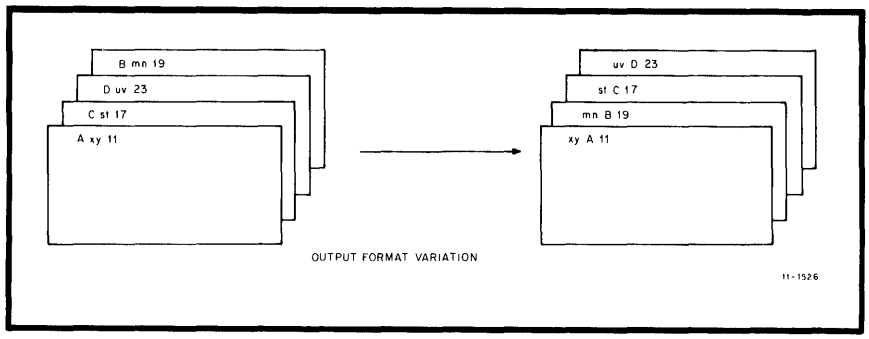
4. Input Format Variation




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.

## 1.2 HOW SORT OPERATES

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. You can write your own control program to take advantage of the separate SORTS subroutines. (See Section 5.4.)

There are three phases of operation in the SORT control program.* In the first phase, SORT reads the command string, decodes, and stores

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

the switch values and the specification file, if present. Any errors in the command string or specification file are reported at this point.

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.

## 1.3 SORT PROCESSING OPTIONS

There are four SORT processes: Record Sort, Tag Sort, Address Routing Sort, and Index 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.

In general, Record Sort performs a relatively slow sort. Tag Sort produces the same kind of output as Record Sort. Tag Sort performs a faster sort if the key size is much smaller than the total record size. The Address Routing and Index Sorts are high-speed sorts that produce compact output. See Table 1-3.

### 1.3.1 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. The first word of a variable length format record contains the size of the record in bytes. This first word is used by the file system 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[1] card reader, console | Slowest |
| SORTT (Tag Sort) | Input | Any | Disk | Slow for large files, large keys |
| | Output | Any | Any appropriate device (including magtape[1]) | |
| SORTA (ADDROUT Sort) | Input | Any | Disk or magtape[1] | Fastest |
| | Output | Fixed, six bytes | Disk | |
| SORTI (Index Sort) | Input | Any | Disk or magtape[1] | Fast |
| | Output | Fixed, 6-byte pointer + original key | Disk | |

[1]Magtape labels are ANSI standard. DOS is not supported for magtape labels.

### 1.3.2 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.

### 1.3.3 Address Routing Sort (SORTA)

SORTA produces address routing files, which consist of relative record pointers, beginning at 1, in binary words. These files can be used as a special index file to access randomly the data in the original file. It is possible to maintain only one data file, but several different index files as needed. Like SORTT, SORTA uses the minimum amount of data necessary in the sorting process. Once the input phase is completed, the input file is not read again. The output data is in a restricted mode. This means that SORTA is the fastest sorting method in the sort package. Do not transfer an ADDROUT index file to a device that cannot handle binary data, such as a printer.

### 1.3.4 Index Sort (SORTI)

SORTI produces an index file (not indexed by RMS-11) consisting of relative record pointers, as in SORTA, and index keys. This makes it slightly slower than SORTA. During processing SORTI handles only the relative record pointers and two forms of the key fields. One form is used for sorting and the other is left as it was in the original data.

NOTE

The SORTI output file contains two extra records at the beginning of the file:

first record - output record count +2

second record - input record count

Each of these additional records contains a key consisting of ASCII nulls. (See Table 1-3.)

If the key size is odd, the key portion of the output record will be padded with a null byte.

### 1.3.5 Graphic Representation of SORTA and SORTI Output

The ADDress ROUTing Sort (SORTA) produces an output file consisting of record indices. Each record index occupies one 6-byte record in the output file. Here is an example of SORTA output:

Let us assume that we are sorting a file consisting of six records with SORTA. If the sequence of record indices corresponding to the sorted records is 5,1,6,3,4,2 then the output from SORTA can be represented as follows:

| RECORD NUMBER | BLOCK NUMBER | | BYTE-IN-BLOCK (2 bytes 16 bits) |
|---|---|---|---|
| | LOW (2 bytes 16 bits) | HIGH (2 bytes 16 bits) | |
| 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 1 6 2 |
| 2 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 0 0 |
| 3 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 2 3 6 |
| 4 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 4 2 |
| 5 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 1 3 2 |
| 6 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 2 6 |

Index Sort (SORTI) produces an output file consisting of record indices plus keys in original form. Each record in the output file consists of a 6-byte record index plus the key in original form. Following is an example of SORTI output.

Assume that you are sorting a file consisting of six records with SORTI, and you are using a key size of four characters (bytes). Note that SORTI writes two additional records at the beginning of the output file: the first is the output record count plus 2; the second is the input record count. The sequence of record indices corresponding to the sorted record, is 5,1,6,3,4,2. The increased output record count in the first additional record reflects the fact that SORTI added two records to the file. Here is part of block 1:

| RECORD NUMBER | BLOCK NUMBER | | BYTE-IN-BLOCK (2 bytes 16 bits) | KEY IN ORIGINAL FORM | |
|---|---|---|---|---|---|
| | LOW (2 bytes 16 bits) | HIGH (2 bytes 16 bits) | | | |
| 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 1 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 |
| 2 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 0 6 | 0 0 0 0 0 0 | 0 0 0 0 0 0 |
| 3 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 1 6 2 | B. A | D. C |
| 4 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | B. A | D. C |
| 5 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 2 3 6 | B. A | D. C |
| 6 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 4 2 | B. A | D. C |
| 7 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 1 3 2 | B. A | D. C |
| 8 | 0 0 0 0 0 1 | 0 0 0 0 0 0 | 0 0 0 0 2 6 | B. A | D. C |

NOTE

ABCD represents the sorting keys in original format. The record count in the first record is an octal 10 (decimal 8).

CHAPTER 2

# THE SORT COMMAND STRING

## 2.1 INTRODUCTION

You initiate the SORT utility program by executing a SORT command string. This chapter describes the command string format for RSTS/E and RSX-11M users. The IAS command string is described in the IAS User's Guide.

## 2.2 HOW TO RUN SORT

### 2.2.1 Running SORT in Interactive Mode

To invoke SORT use one of the following commands after the system prompt:

```
SRT
RUN  [1,2] SORT
RUN$SORT
```

SORT then prints SRT> to indicate that it is ready to accept a command string. You must specify one input file and one output file. A second input file, if you give one, is the specification file.

One level of indirect command files is permitted. An example SORT command string is shown below:

```
SRT>OUTPUT=INPUT/FO:VAR:100,SPEC
```

This command string directs SORT to sort the file INPUT.DAT on the system device and place the sorted file OUTPUT.DAT on the system device. The specification file SPEC.SRT on the system device is used to define the sorting parameters. The switch /FO:VAR:100 is described in Section 3.1.7.

After you enter the SORT command string, SORT begins its processing. If there are errors in the command string or specification file, they are immediately reported on the console. At the end of a successful run, SORT prints the following message:

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

This message tells you the real or wall clock time necessary to perform the sort and the number of records in the output file.

When the sort is done, SORT again prompts with SRT>. At this point you can either enter another command string or terminate execution with ^Z (CTRL/Z) (or ^C - RSTS/E only).

### 2.2.2 Running SORT in RSTS/E Batch Mode

To run SORT in RSTS/E batch mode, precede your command string with the following:

```
$RUN $SORT
$DATA
```

The command string is the same as that described in the preceding section.

## 2.3 THE COMMAND STRING FORMAT

The SORT program requires a command string to direct each job. The command string names the input and output files and the devices on which they reside. Command string switches define the input and output file parameters. The specification file or additional command string switches control the sort. The use of the switches and the specification file is explained in Sections 3.1 through 3.3.

A command string has one of the following forms:

```
output-files-list=input-files-list

@command-file
```

Both output-files-list and input-files-list are strings consisting of file specifications separated by commas. Each file specification selects a file to be used as an output or input file by the system program.

In particular, the SORT output-files-list consists of one file specification; and the SORT input-files-list consists of one or two file specifications. The second input file is optional, and is described in Section 3.2 as the specification file.

A file specification may have one or more switches associated with it. A switch is an indicator used to select program options. Switches for SORT have the following form:

```
/name:value
```

The switch name is a string of characters of which only the initial two are significant.

The second form of command string specifies an indirect command file. Each line on the file is used as a command string (as if you had typed it at the terminal) before any command strings following the file specification are accepted.

The command string syntax is described more completely in the Programmer's Reference Manual for your system or, for RSTS/E users, the RSTS/E User's Guide.

### 2.3.1 File Specifications

A file specification has the following form:

```
dev:[uic]filenam.typ;ver/sw1:v1:...:vn/sw2:v1...
```

where:

| | |
|---|---|
| dev: | 2-character alphabetic legal device code and optional 1- or 2-digit unit number. The default is SY, the system disk. |
| [uic] | project-programmer number specifying the directory. The UIC is optional, but, if used, must be a valid User Identification Code. |
| filename | any 1- to 6-character (RSTS/E only) or 1- to 9-character alphanumeric string that specifies a file name. |
| .typ | 1- to 3-character file type. System programs default the file type to an appropriate standard type. So, the typical user will not need to specify the file type. |
| ;ver | version number of the file (for RSX-11M and IAS users only -- omit for RSTS/E). This can normally be omitted because the system defaults to the highest existing version number on input and to the highest existing version number plus 1 on output. |

File types default as shown below:

| File | Default Type |
|---|---|
| Input | DAT |
| Output | DAT |
| Specification | SRT |
| Command | CMD |

File specifications are discussed in detail in the Programmer's Reference Manual or, for RSTS/E users, the RSTS/E User's Guide.

Table 2-1
Device Specification Codes

| Dev: | Device* |
|---|---|
| SY: | System disk, default device |
| DKn: | Disk, cartridge unit n; n=0 is the default condition |
| TI: | User terminal |
| CR: | Card reader |
| LP: | Line printer |
| TT0: | Operator console |
| DB: | RPJ04/RPJ06 disk |
| DP: | RP11/RP03 disk |
| DM: | RK06 disk |
| MM: | Magnetic tape |
| MT: | Magnetic tape |

* See your System User's Guide for specific device types.

CHAPTER 3

# CONTROLLING SORT 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. RMS requirements have made it necessary to completely revise the SORT switches. These changes have been reflected in the switches described here. All numeric switch values are decimal, not octal. All switches are valid for both input and output files and on all systems 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. For index and address routing sort processes, the default size is based on the number of records sorted. See Section 5.1.4 for details.

### 3.1.2 BLOCKSIZE (/BL:n)

The BLOCKSIZE switch is valid for magtape files only and is used to specify the 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 | Build value | Input |
| FILES | /FI:n | Build value or 5 | Both |
| FORMAT | /FO:x:n | (See footnote) | Both |
| INDEXED SEQUENTIAL | /IN:x | (See footnote) | Input |
| KEY | /KE:abm.n | C, N, first position of field, length of field | Input |
| PROCESS | /PR:x | R (record sort) | Input |
| RELATIVE | /RE | (See footnote) | Output |
| SEQUENTIAL | /SE | (See footnote) | Output |
| SIZE | /SI:n | RSTS clustersize or RSX-11M retrieval window size (see Section 3.1.13) | Output |

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

The default for output files is SEQUENTIAL. For SORTR and SORTT, 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. For SORTI and SORTA, the output record format will be FIXED.

Default record sizes are six bytes for SORTA, six bytes plus the size of the index for SORTI, and the size of the input record for SORTR and SORTT.

### 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). See Section 5.1.6 for details.

### 3.1.6 FILES (/FI:n)

The FILES switch specifies the maximum number of intermediate scratch files. Legal values are from 3 to 8. If you patched the location FILES at task build time, the default value now is the one you specified then. If you did not use a patch, 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. If you have not used SORT before, refer to Appendix B and become acquainted with the concepts of zones and digits.

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

B 2'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

2. If the characters represent a standard FORTRAN IV 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 PROCESS (/PR:x)

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

R - RECORD (the default)
T - TAG
A - ADDRESS ROUTING
I - INDEX

### 3.1.11 RELATIVE (/RE)

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

### 3.1.12 SEQUENTIAL (/SE)

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

### 3.1.13 SIZE (/SI:n)

The SIZE switch specifies the file cluster size (RSTS) or the size of the retrieval window (RSX-11M). The ranges and defaults for this switch will be determined by your operating system. Check with your system manager for this information and record the ranges and default below:

SYSTEM:_____ RANGE:_____ to:_____ DEFAULT:_____

## 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, in card form, two possible arrangements of the specification file whose elements are explained below.

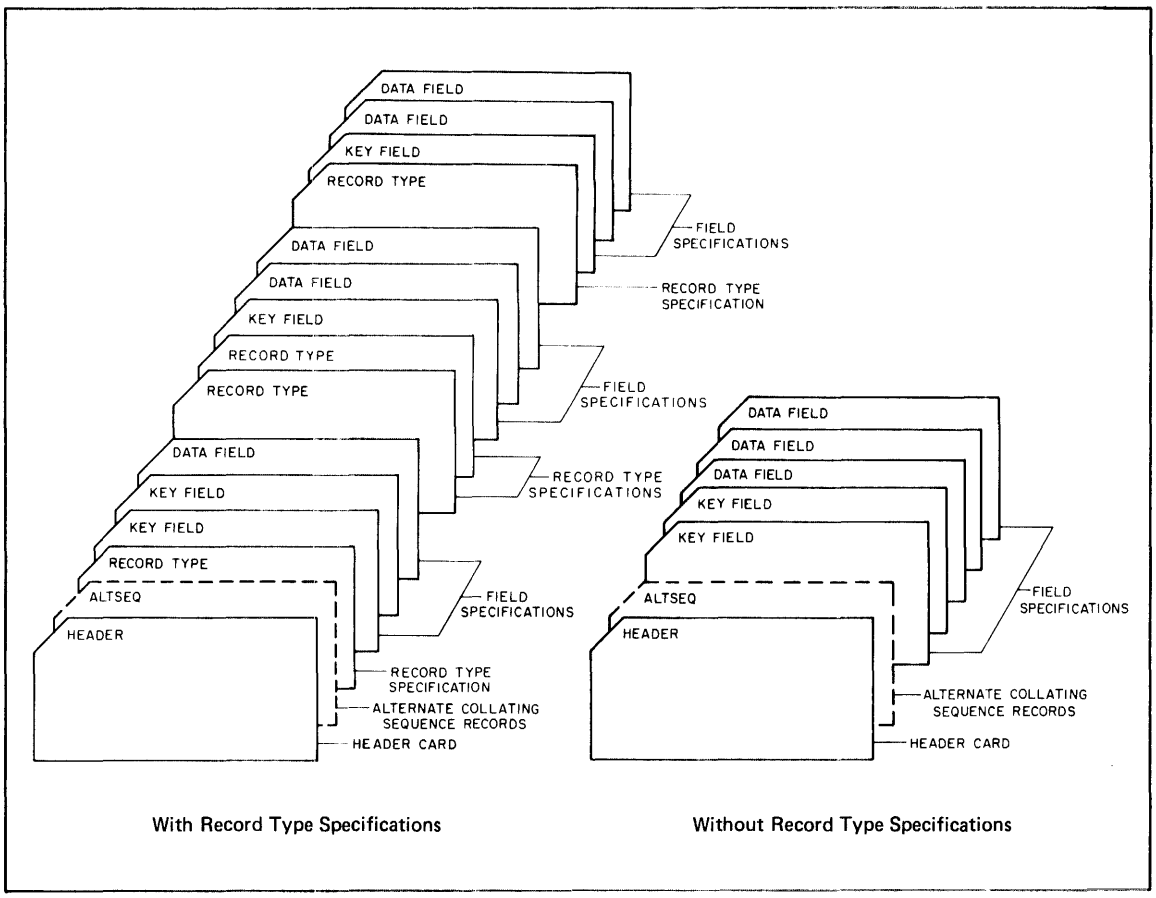



Figure 3-1 Specification File Format

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 job.

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. Data fields are not specified for SORTA or SORTI because the output file contains only pointers and possibly some restricted-format key data.

The format of each type of specification record is fixed. The SORT specification form, based on card columns, 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 |

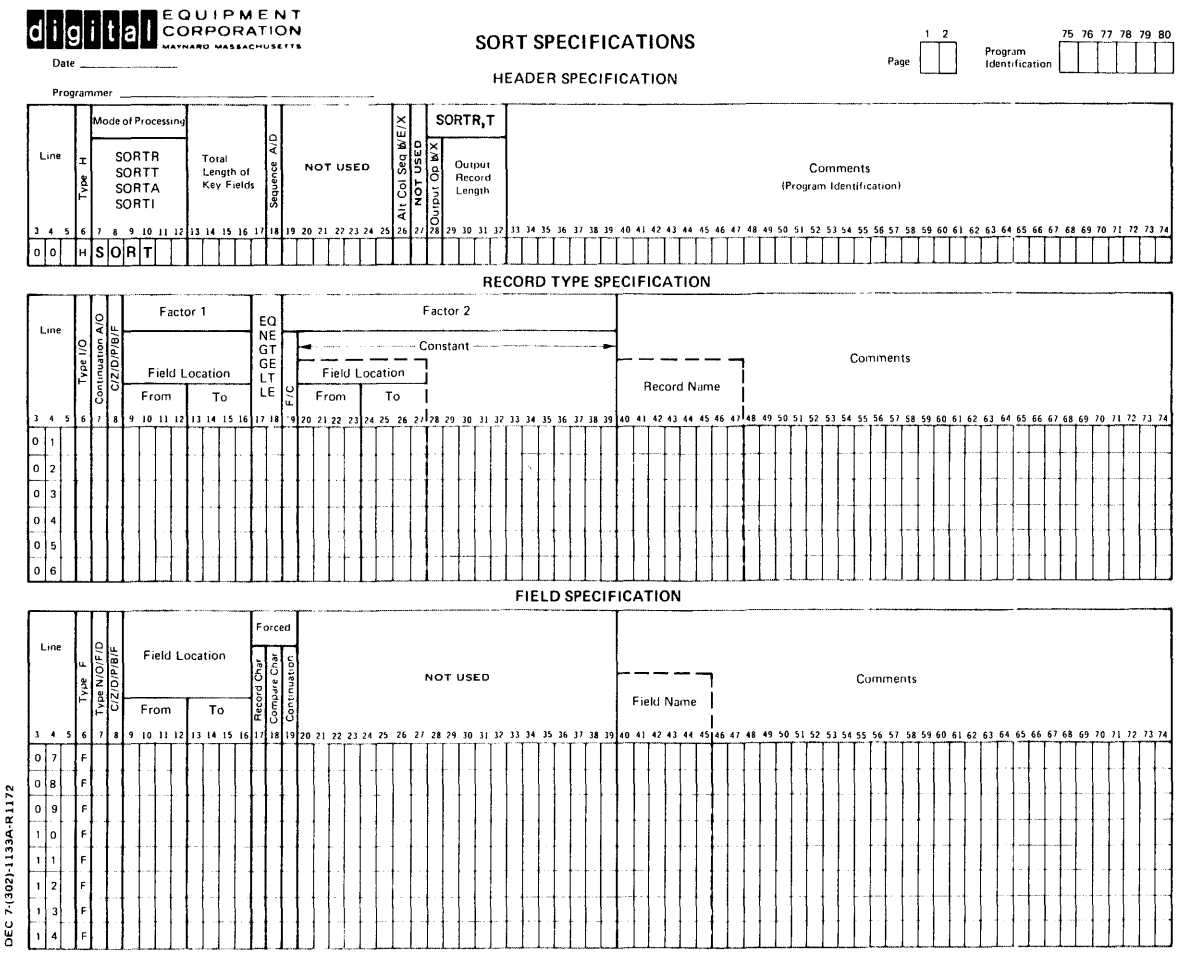



Figure 3-2 SORT Specification Form

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<br><br>Legal value: H |
| 7 - 12 | Type of SORT (must be left-justified)<br><br>Legal values:<br><br>SORTR or blanks - Record SORT<br>SORTT - Tag SORT<br>SORTA - ADDROUT SORT<br>SORTI - Index SORT |
| 13-17 | Total of all key field sizes<br><br>Legal values: 1 - 16383<br>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 sequence<br><br>Legal values:<br><br>A or blank - ascending<br>D - descending |
| 19 - 25 | (not used) |
| 26 | Alternate collating sequence<br><br>Legal values:<br><br>blank - standard ASCII sequence<br>E - EBCDIC sequence<br>X - user-modified ASCII sequence |
| 27 | (not used) |
| 28 | Output option for SORTR and SORTT ONLY<br><br>Legal values:<br><br>X - key fields created by SORT will be stripped from the output record<br>blank - no action |
| 29-32 | Output record length for SORTR and SORTT<br><br>ONLY<br><br>Legal values:<br><br>decimal number equal to the number of bytes in the largest output record |

| Columns | Explanation and Legal Entries |
|---|---|
| 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.**<br><br>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) |

| Columns | Notes |
|---|---|
| 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<br><br>Legal values:<br><br>I - Include the records described in this line in the sort<br><br>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)<br><br>Legal values:<br><br>A - logical AND<br>O - logical INCLUSIVE OR |
| 8 | Field mode (defines the processing to be applied to the data field defined)<br><br>Legal values:<br><br>B - binary<br>C - alphanumeric<br>D - numeric<br>F - binary floating point in 2 to 4 words<br>I - numeric with leading and separate sign<br>J - numeric with trailing and separate sign<br>K - numeric with sign leading and overpunched in first byte<br>P - packed decimal<br>Z - zone, see Table B-2, Appendix B |

| Columns | Explanation and Legal Entries |
|---|---|
| 9 - 12 | Factor 1 (position of the base comparison factor is in the first byte; the first byte in the record is byte 1)<br><br>Legal values:<br><br>a decimal number - location of the first byte<br><br>blanks - specifies a single byte field |
| 13 - 16 | Factor 1 endpoint<br><br>Legal values:<br><br>a decimal number - location of last or only byte in the field |
| 17 - 18 | Relationship of Factor 1 to Factor 2<br><br>Legal values:<br><br>EQ - equal<br>NE - not equal<br>LT - less than (i.e. F1 LT F2)<br>GT - greater than<br>LE - less than or equal to<br>GE - greater than or equal to |
| 19 | Comparison factor format (indicates whether Factor 2 is a field or a constant)<br><br>Legal values:<br><br>F - identifies a field<br>C - identifies a constant |
| 20 - 39 | Factor 2 (This takes two different formats depending on whether Factor 2 is a field or a constant.)<br><br>If Factor 2 is a field:<br><br>20 - 23 location of first byte - a decimal number or blanks as for Factor 1.<br><br>24 - 27 location of the last or only byte - a decimal number as for Factor 1.<br><br>28 - 39 (not used)<br><br>If Factor 2 is a constant:<br><br>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:<br><br>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:<br><br>B AND C OR A<br><br>SORT will react:<br><br>(B AND C) OR A,<br><br>which gives the required True value.<br><br>On the other hand,<br><br>A OR B AND C<br><br>gives a True value in both the SORT and non-computerized logic as it stands. |

| Columns | Explanation |
|---|---|
| 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 Fortran IV 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, if the sorting process is SORTR or SORTT, 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. The 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<br>Legal values:<br>N - key field, normal sort sequence (see position 18 of Header)<br>O - key field opposite sequence<br>F - key field, special treatment required (see positions 17 - 19)<br>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 (q.v.). |
| 9 - 12 | Field location - location of first byte of multi-byte field<br>Legal values:<br>a decimal number - location of first byte of field<br>blanks - specifies one-byte field |
| 13 - 16 | Field location - last or only byte<br>Legal values:<br>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:<br>Legal values:<br>blank - the force always occurs<br>any other character - the force occurs only if this character is found |
| | 18 Replacement character<br>Legal values:<br>any legal character |

| Columns | Explanation and Legal Entries |
|---|---|
| 19 | Continuation character<br><br>Legal values:<br><br>blank - this is a new forced key<br><br>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.

NOTE

SORT rejects forced key specifications for SORTI.

### 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.

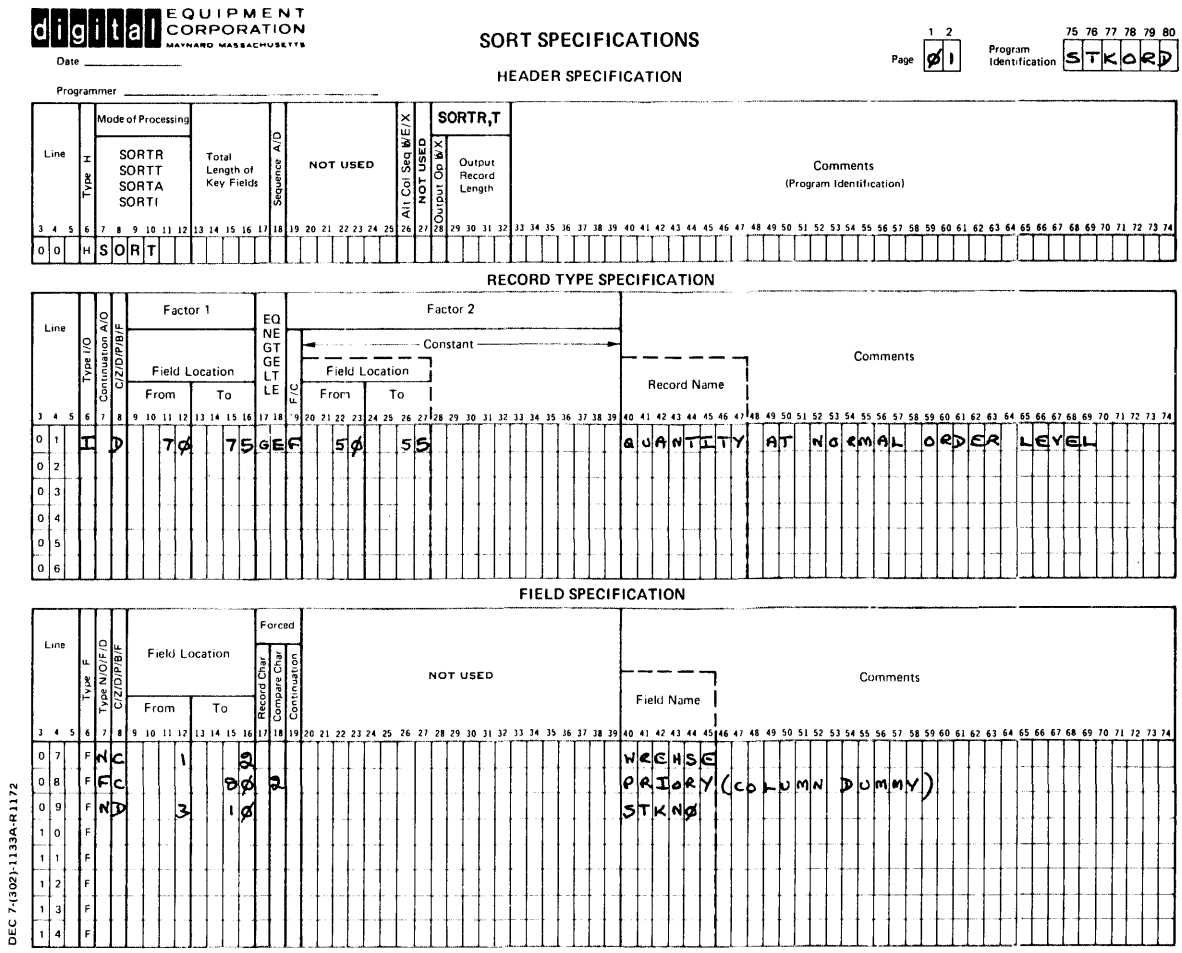



Figure 3-3 SORT Specification Entries

## 3.3 SORT SPECIFICATIONS SUMMARY

Table 3-2
Header Specifications

| Column | Entry | Explanation |
|---|---|---|
| 6 | H | Header Specification |
| 7-12 | SORTR<br>SORTT<br>SORTA<br>SORTI | Record Sort<br>Tag Sort<br>ADDROUT Sort<br>Index 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<br>D | Processing sequence: Ascending or Descending |
| 26 | Blank<br>E<br>X | ASCII collating sequence<br>EBCDIC collating sequence<br>ASCII with modification listed in following lines |
| 28 | Blank (SORTR, T only) | 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 (SORTR, T only) | 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 | C | Character - the field in columns 9-16 (Factor 1) is alphameric |
| | Z | Zone - use the zone value for the entry |
| | D | Digit - use the digit value or convert the FORTRAN IV number to binary |
| | 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 |
| | B | Binary - the field contains 2's complement binary data |
| | F | Floating - treat the field as a 2- or 4-word floating binary representation |

(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 | C | Character - alphanumeric key or data field |
| | Z | Zone - zone value |
| | D | Digit - use digit value or convert to binary for FORTRAN IV numbers |
| | 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 |
| | B | Binary - the key field is in 2's comple-ment binary notation |
| | F | Floating - the key field is in 2- or 4-word floating binary representation |
| 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

There are three main categories of errors that SORT can detect:

- 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:<br><br>C - command decoder<br>M - merge<br>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 <CR>. "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. If you need more space, see Appendix D.

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, SORTA, SORTI, 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 if type of SORT is SORTR or SORTT.

   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.

   b. There are no more clusters of the size you specified (RSTS/E only).

5. INAPPROPRIATE FILE ORGANIZATION

   a. /IN switch was not specified.

   b. You tried to access an indexed file and RMS-11K support was not linked in.

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 (see Chapter 5). They are reported as:

SORT ERROR -- CODE nn

where nn is the code number returned by a SORTS subroutine (described in paragraph 5.3).

## 4.7 RMS ERROR CODES

A complete listing of RMS status codes can be found in Appendix G. 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 occur only under RSX-11M, and 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 RSX-11M

| 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 | Files-11 ACP Enter function failure |
| -624 | File extension failure |
| -1088 | File could not be marked for deletion |
| -1456 | Files-11 ACP 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 may occur under any operating system. They 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 0 or is too large (indexed file) or is not 4 (relative file) |
| -1168 | Not enough room to open an indexed file |

Status code -864 occurs during subroutine usage only. It indicates that you did not initialize RMS before control passed to SORT.

Table 4-3 gives other status codes that commonly occur.

Table 4-3
Other Commonly Occurring Status Codes

| Code | Brief Explanation |
|---|---|
| -112 | Variable length records on ANSI-labelled magtape are not ANSI D format |
| -336 | CLOSE function failed (RSTS/E only) |
| -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 G 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 0 during $CREATE |
| -1152 | Not at end of file |
| -1200 | Open function failed (RSTS/E only) |

(Continued on next page)

Table 4-3 (Cont.)
Other Commonly Occurring Status Codes

| Code | Brief Explanation |
|---|---|
| -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 (not a RSTS/E occurrence) |
| -1784 | Device is write-locked |

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

CHAPTER 5

# INTERNAL OPERATION

## 5.1 FILES

### 5.1.1 User and Scratch File Name Conventions

Input, output, and specification files may have any name acceptable to the operating system.

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

### 5.1.2 LUN Assignments

The following are the default Logical Unit Number assignments as reflected in the task build command file SRTxxx.CMD (xxx is system dependent - see Section F.2):

| LUN | Device | Use | Internal Name |
|---|---|---|---|
| 1 | TI: | Command Input File | - |
| 2 | TI: | Message Input File | SORT:MSGLUN |
| 3 | SY: | Input File | SORT:INLUN |
| 4 | SY: | Output File | SORT:OUTLUN |
| 5 | SY: | Specification File | SORT:XLUN |
| 6 | SY: | First Scratch File | SORTS:$RFIRL |
| 7-13 | SY: | Other Scratch File | - |

To change the internal correspondence between LUN and usage, include the following statement in the command file:

```
GBLPAT=x:y:z
```

where:

x:y is the internal name, such as SORT:MSGLUN.
z is the new LUN in octal.

To change the devices to LUN correspondence, change the following statement in the command file:

```
ASG=x:y
```

where:

x is the device name.
y is the LUN in decimal

### 5.1.3 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.

### 5.1.4 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. Under the index and address routing processes, SORT calculates a size for the output file by multiplying the size of the output record by the number of records sorted. 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 you are using the tag or record sorting process and 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. Read Section D, especially Section D.3, again carefully, and remember that there is a tradeoff in this situation between size and speed. 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.

### 5.1.5 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). Thus, bits 14 and 15 are always 0. |
| 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 and SORTA, it is a 2-word relative record pointer. This pointer is joined by the key fields in their original form during a SORTI. |

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

### 5.1.6 Scratch Files Use

The scratch files switch, FILES (see Section 3.1.5), 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

---

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

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. The default number is allocated if the FILES switch is unspecified. This default is defined at task building time via "GBLPAT=SORT:FILES:X" (3< X <10 octal). The SORTS routines use an unbroken ascending series of LUNs starting at the value in internal location $RFIRL and continuing for the number of files specified in the internal location FILES (the default is 5).

Depending upon the sorting process and/or the sorting specifications, each scratch file could require half again as much disk storage as the original input file. Take this into consideration when assigning the scratch file LUNs to disk devices. Also, the executive's files system functions most efficiently if few blocks are occupied and there is minimal space fragmentation of the disk space. Use private disks if you can. This will speed up the sort, especially if you put all your scratch files on private disk space and the input and output files on public disks. If you have more than one disk, scatter the LUNs among them so that no two successive LUNs are on the same disk. Use the fastest disks for scratch files.

## 5.2 ORGANIZATION OF SORT

As noted in Section 1.2, the SORT program goes through three phases. Each phase is actually set up as an overlay with a resident control program to handle the overlays in the correct sequence.

Here are the names and functions of the SORT modules:

1. RSORT initializes the system, then serves as the resident error processor for the remaining modules.

2. SORTC decodes the Command String and Specification File. This routine is the first section to be called by RSORT. After SORTC finishes, it is overlaid by SORTP.

3. SORTP handles the pre-sort phase based upon data transmitted by SORTC and is associated with appropriate subroutines from SORTS. It is replaced by SORTM when the pre-sort is done.

4. SORTM controls the main sort and output. It recalls SORTC for a new sort when it has finished its task.

5. SORTS is a collection of subroutines that actually do the sorting.

## 5.3 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 G 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, SORTA, SORTI. |

## 5.4 USING THE SORTS SUBROUTINE PACKAGE

You can associate the SORTS subroutine package with your own control program to perform a particular type of sort, provided that the control program meets the necessary interface requirements.

### 5.4.1 Conventions and Standards

All of the entry points in the SORTS subroutines are entered via the coding:

```
          MOV     #TAG,R5
          JSR     PC,X
            .
            .
            .
     TAG: .BYTE   N,0
          .WORD   ARGUMENT-1
            .
            .
            .
          .WORD   ARGUMENT-N
```

where:

| | |
|---|---|
| PC | is Register 7, |
| X | is an entry label, |
| N | is the number of arguments. |

All of the sort subroutines require arguments which are word-bound (16-bit word addresses). This is because the routines use word oriented instructions, rather than byte oriented instructions, for speed advantages.

A detailed description of the calling procedure for each routine follows. In all calling procedures, IERROR is the starting address of a user-generated error processor to be used if the call aborts.

#### 5.4.1.1 RSORT - Initializing the Sort

```
          MOV     #TAG,R5
          JSR     PC,RSORT
            .
            .
            .
     TAG: .BYTE   7(11.),0;7 (OPTIONALLY 11) ARGUMENTS TO RSORT
          .WORD   <ADDRESS OF ERROR CODE>
          .WORD   <ADDRESS OF NUMBER OF BYTES IN ALL KEYS>
          .WORD   <ADDRESS OF NUMBER OF BYTES IN LARGEST RECORD>
          .WORD   <LOCATION OF MOST MAJOR KEY WORD>
          .WORD   <FIRST LOCATION IN WORK AREA>
          .WORD   <ADDRESS OF SIZE OF WORK AREA IN BYTES>
          .WORD   <ADDRESS OF NUMBER OF SCRATCH FILES>
          -----O P T I O N A L L Y -----
          .WORD   <ADDRESS OF NUMBER OF BUFFERS PER FILE>
          .WORD   <ADDRESS OF CLUSTERSIZE FOR RSTS/RETRIEVAL POINTER
                  FOR RSX>
          .WORD   <ADDRESS OF PRIMARY ALLOCATION FOR SCRATCH FILES>
          .WORD   <ADDRESS OF FIRST SCRATCH FILE LUN>
```

For FORTRAN, the call is as follows:

```
CALL RSORT (IERROR,KEYSIZ,MAXREC,KEYADR,IWRKLO,IWKSIZ,IFILES
          [,BIGBUG,SCRCLF,PRIALQ,FIRLUN])
```

where:

| | |
|---|---|
| KEYSIZ | is the decimal byte count of the total key size. KEYSIZ must be even (incremented by 1 if not) and positive (error 12 if not). |
| MAXREC | is the decimal size in bytes of the largest record in the input file. MAXREC must be even (incremented by 1 if not) and positive (error 13 if not). The sum of KEYSIZ and MAXREC cannot exceed 16,383 (error 16 if greater). |
| KEYADR | is the address of the most significant word in the major key. KEYADR must be a word-bound address (error 14 if not). See Section 5.4.2 for details. |
| IWRKLO | is the address of WORKLO, the first word in the SORT work area. IWRKLO must be word-bound (incremented by 1 if not). See Appendix D, especially Figure D-1a. |
| IWKSIZ | is the size of the work area in bytes (decimal). |
| IFILES | is the number of scratch files to be allocated. IFILES must be greater than 2 and less than 11 (errors 17 and 20 respectively if not). |
| BIGBUF | is the number of 512-byte buffers to be allocated for each scratch file and the number of contiguous blocks read or written during each scratch file read or write. The default is 1. For details see Appendix D. |
| SCRCLV | is the RSTS/E clustersize or the RSX-11M number of retrieval pointers. The default is 0. For details see Appendix D. |
| PRIALQ | is the decimal number of blocks to be allocated to each scratch file at open time. For details see Appendix D. |
| FIRLUN | is the first Logical Unit Number to be used by the scratch files. The SORTS subroutines use the set of LUNs running consecutively from FIRLUN to (FIRLUN + IFILES - 1) in octal representation. The default is 5. |

FORTRAN users should remember to designate BIGBUF, SCRCLV, PRIALQ, and FIRLUN as INTEGERS.

#### 5.4.1.2 RELES - Passing Input Records to the Sort

```
        MOV     #TAG,R5
        JSR     PC,RELES
          .
          .
          .
TAG:  .BYTE   3,0 ;3 ARGUMENTS TO RELES
      .WORD   <ADDRESS OF ERROR CODE>
      .WORD   <ADDRESS OF THE SIZE OF THE RECORD, IN BYTES>
      .WORD   <LOCATION OF THE FIRST WORD OF THE RECORD>
```

For FORTRAN, the call is as follows:

```
CALL RELES(IERROR,IRECSZ,RECADR)
```

where:

IRECSZ is the decimal byte count of the size of the record to be released. IRECSZ must be positive and less than or equal to MAXREC (error 5 if not) (See Section 5.4.1.1).

RECADR is the address of the record (error 15 if not word-bound).

#### 5.4.1.3 MERGE - Merging the Scratch Files

This is called after all input has been read.

```
        MOV    #TAG,R5
        JSR    PC,MERGE
          .
          .
          .
TAG:  .BYTE  1,0 ;1 ARGUMENT TO MERGE
      .WORD  <ADDRESS OF ERROR CODE>
```

For FORTRAN the call is:

```
CALL MERGE(IERROR)
```

#### 5.4.1.4 RETRN - Requesting an Output Record

```
        MOV    #TAG,R5
        JSR    PC,RETRN
          .
          .
          .
TAG:  .BYTE  3(4),0; 3 (OPTIONALLY 4) ARGUMENTS TO RETURN
      .WORD  <ADDRESS OF ERROR CODE>
      .WORD  <ADDRESS OF WORD TO CONTAIN RECORD SIZE>
      .WORD  <LOCATION OF FIRST WORD OF THE RECORD>
      ---------- OPTIONALLY ----------
      .WORD  <ADDRESS OF WORD TO CONTAIN LOCATION OF RECORD>
```

For FORTRAN, the call is as follows:

```
CALL RETRN(IERROR,IRECSZ,RECADR[,LOCREC])
```

where:

IRECSZ is the decimal byte count of the size of the returned record.

RECADR is the address of buffer to receive record (error 15 if not word-bound).

LOCREC if RECADR is 0, RETRN places the address of the record in the location rather than actually moving the record.

#### 5.4.1.5 ENDS - Ending the Sort

```
        MOV     TAG,R5
        JSR     PC,ENDS
          .
          .
          .
TAG:  .BYTE 1,0 ;1 ARGUMENT TO ENDS
      .WORD  <ADDRESS OF ERROR CODE>
```

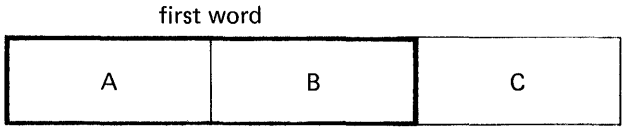
For FORTRAN the call is:

```
CALL ENDS(IERROR)
```

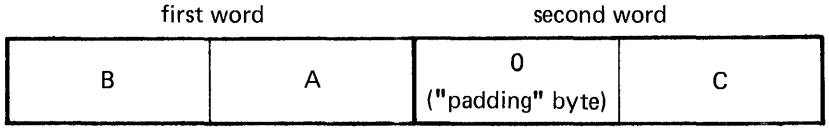
### 5.4.2 Setting up the Keys

Before calling RELES, you must first set up the key area, usually outside the record itself. This area must be contiguous and an even number of bytes long. Round up if odd.

If you are using byte data, the bytes within each word of the key must be reversed in order before the key is stored. If your key is an odd number of bytes, a padding byte of 0s must be added to your key before the bytes of the final word are reversed. Thus, a key that looks like this in the record:




looks like this when it is stored:




If you are using word data, you must move the data word by word into the key area, starting at the highest key location and working downward.

The address passed to RSORT in the fourth parameter is the address of the most major WORD. SORT always does word compares, starting at the word in the highest memory location, and, if necessary, working down to the least significant byte in the lowest memory location.

The address to be used for KEYADR can be figured as follows: assume KEYSTR is the first word of the key area which is KEYSIZ bytes long. Then KEYADR = KEYSTR+KEYSIZ-2.

NOTE

The comparison is logical, so all 16 bits of a word are significant. There is no implied sign. The control program must therefore organize the key data passed to SORTS in a form that ensures the correct sequence, even if the original data may be algebraic.

### 5.4.3 Calculating the Size of the Work Area

The size of the work area, IWKSIZ, must be a certain minimum calculated as follows.

```
IWKSIZ = IFILES * (512. * BIGBUF + (100 + MAXREC + KEYSIZ + 10.))
```

If you supply space below this minimum, SORTS will keep decreasing the number of files until either the above equation is satisfied or the number of files drops below 3, which will then cause RSORT to return with error code 17.

Any extra core will be used to expand the in-core sort area. Thus, in general, the more space supplied, the faster the sort.

### 5.4.4 Typical Calling Sequences

To sort the file IN.TXT to produce the file OUT.TXT:

1. Open IN.TXT.
2. Call RSORT to initialize the sort.
3. Read the next logical record from IN.TXT. If no more data, go to Step 6.
4. Set up the keys from the new record.
5. Call RELES to give the record to the sort, then loop back to Step 3.
6. Close IN.TXT.
7. Call MERGE to collate the data.
8. Open OUT.TXT.
9. Call RETRN to get the next sorted record. If no more records, go to Step 11.
10. Write the record onto the output file; loop to Step 9.
11. Close OUT.TXT.
12. Call ENDS to clean up the sort scratch files.

NOTE

The routine RETRN indicates "no more records" by returning a negative value in the error code.

### 5.4.5 Linking SORTS Subroutines with Your Program

The actual sorting subroutines are contained in SORTS.OBJ and SIORMS.OBJ. You can link these to your own calling program using Task Builder. For example, if your program name was PROG:

```
TKB PROG=PROG,SORTS,SIORMS[,[1,1]RMSLIB/LB]
```

# INTRODUCTION TO THE APPENDICES

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. You will find this 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 to 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 discusses SORT internal operations. This is information for the benefit of the advanced user and is not essential for successful SORT use.

Appendix E guides your installation of SORT.

Appendix F lists RMS status codes.

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

## PRINTABLE CHARACTERS

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 |
| | 1 | 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.

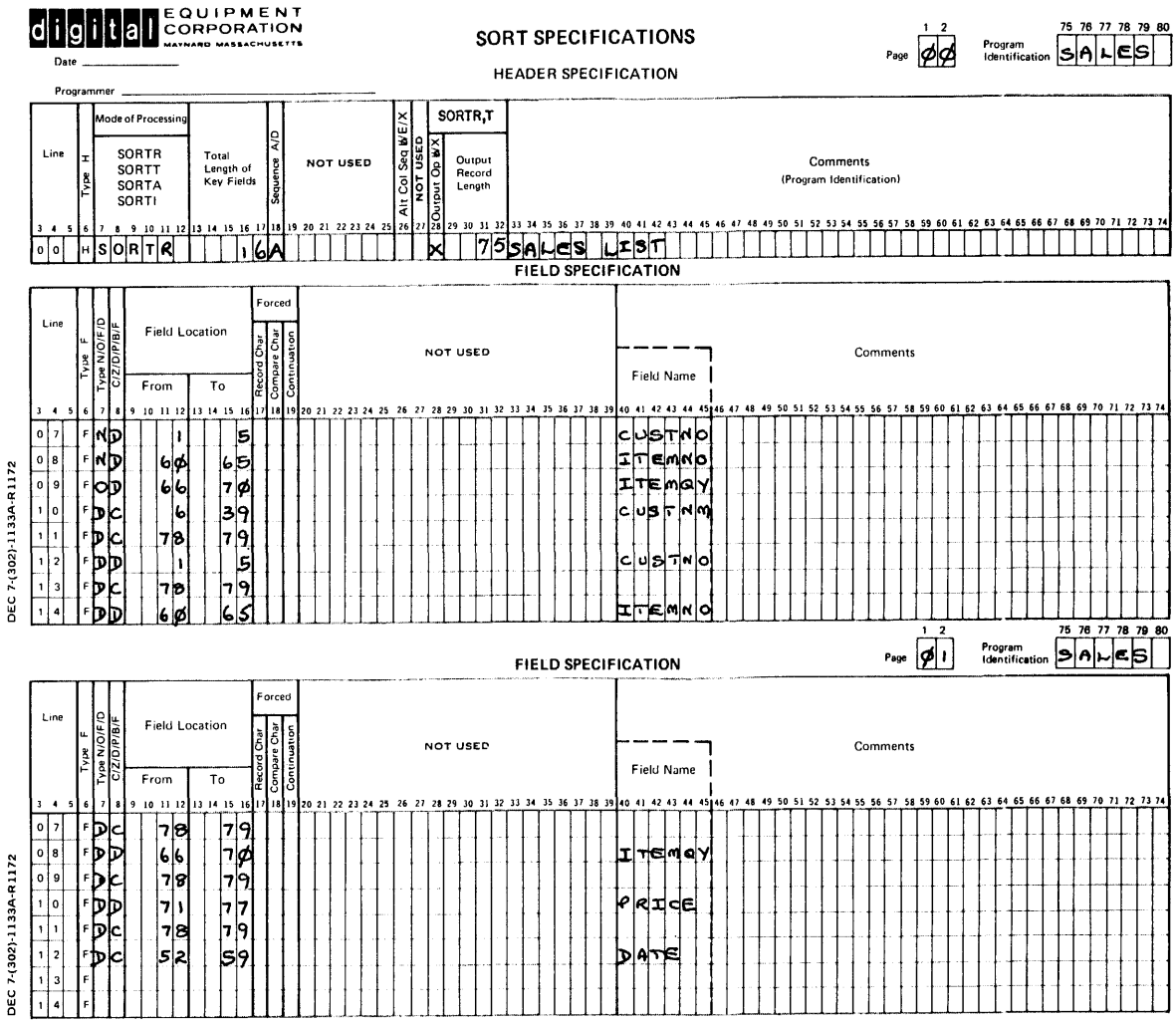



Figure C-1 Sales List

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.

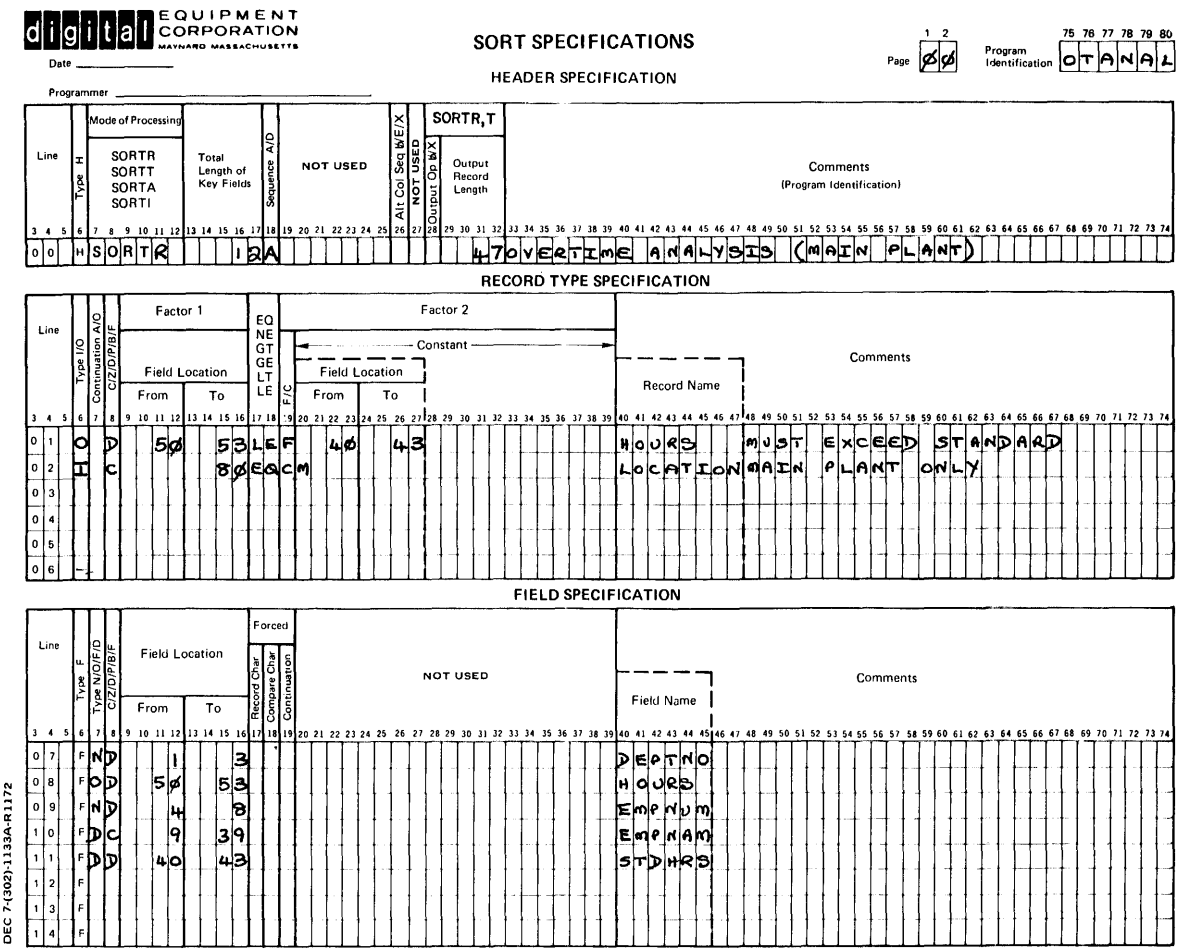



Figure C-2 Overtime Analysis

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.

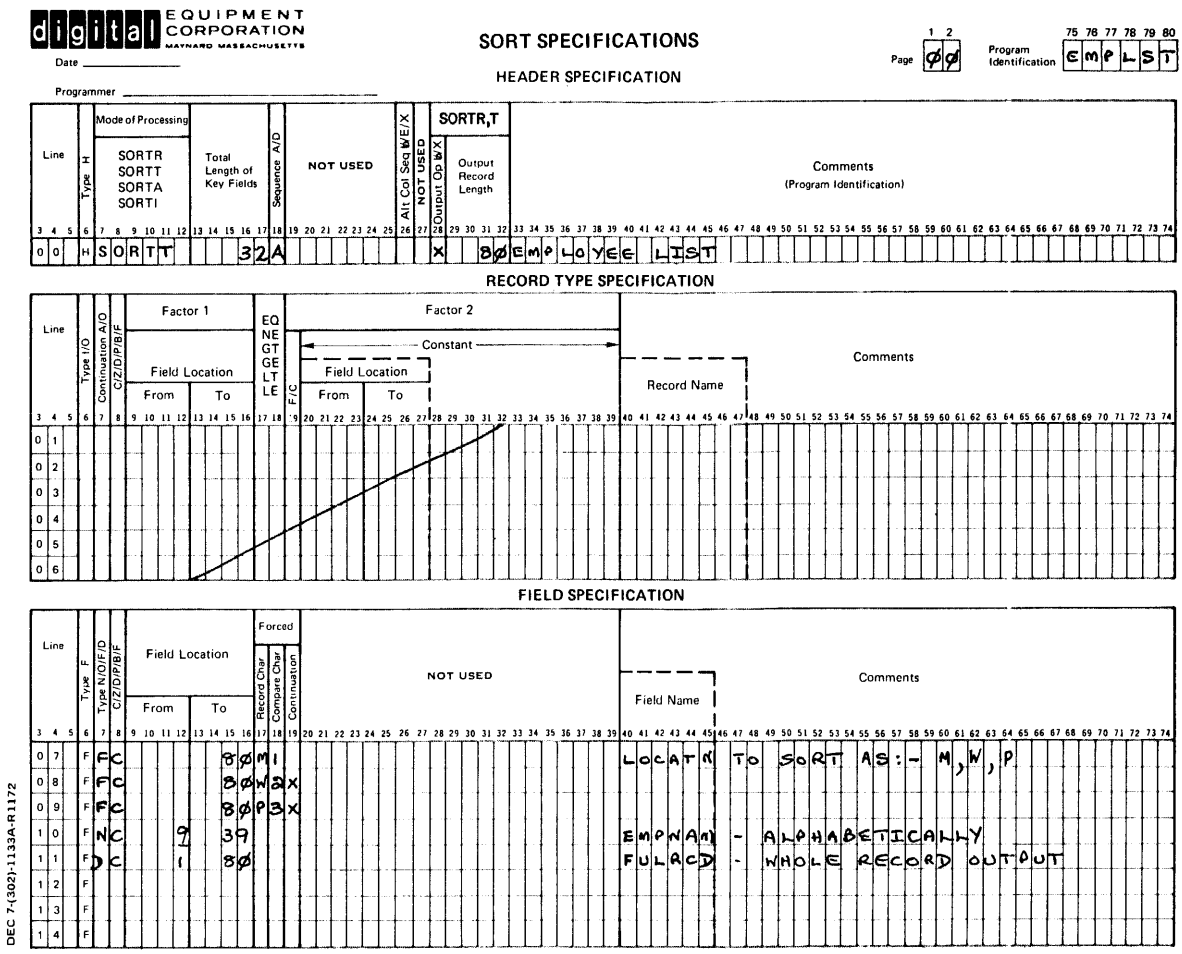



Figure C-3 Employee List

EXAMPLE 4: BONUS ANALYSIS

This Index Sort illustrates the use of multiple Include and Omit lines. The first category of Omit lines excludes all main plant industrial employees' records that do not show bonus eligibility; the second set of Omit lines excludes any employees' records from any other plant that do not show bonus eligibility. The line with an I in column 6 and no other entries includes all other records; i.e., the records of industrial staff employees from the main plant and all employees from any other plant who are eligible for a bonus.

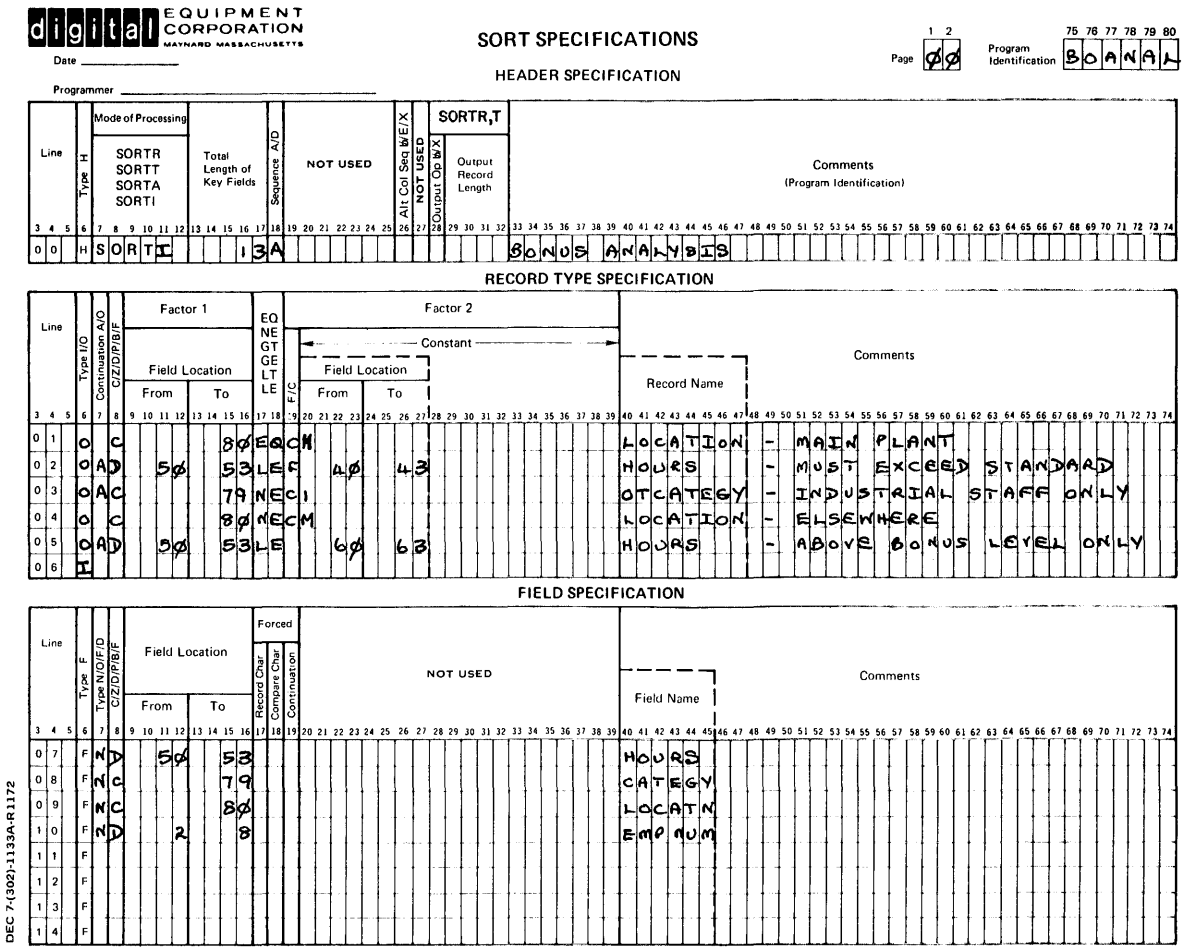



Figure C-4 Bonus Analysis

EXAMPLE 5: STOCK ORDER LIST

1-2 Warehouse Identification Code
3-10 Stock Number
50-55 Quantity of Items Required to be in Stock
70-75 Quantity of Items in Stock

This is an ADDROUT Sort intended to produce a file of stock numbers by warehouse. Those records whose stock quantity level is lower than the accepted level are listed first in each warehouse category. A priority number (1 for under-stocked items; 2 for items of sufficient quantity) is forced into position 3 of the output record.

The output file records will be in the following order:

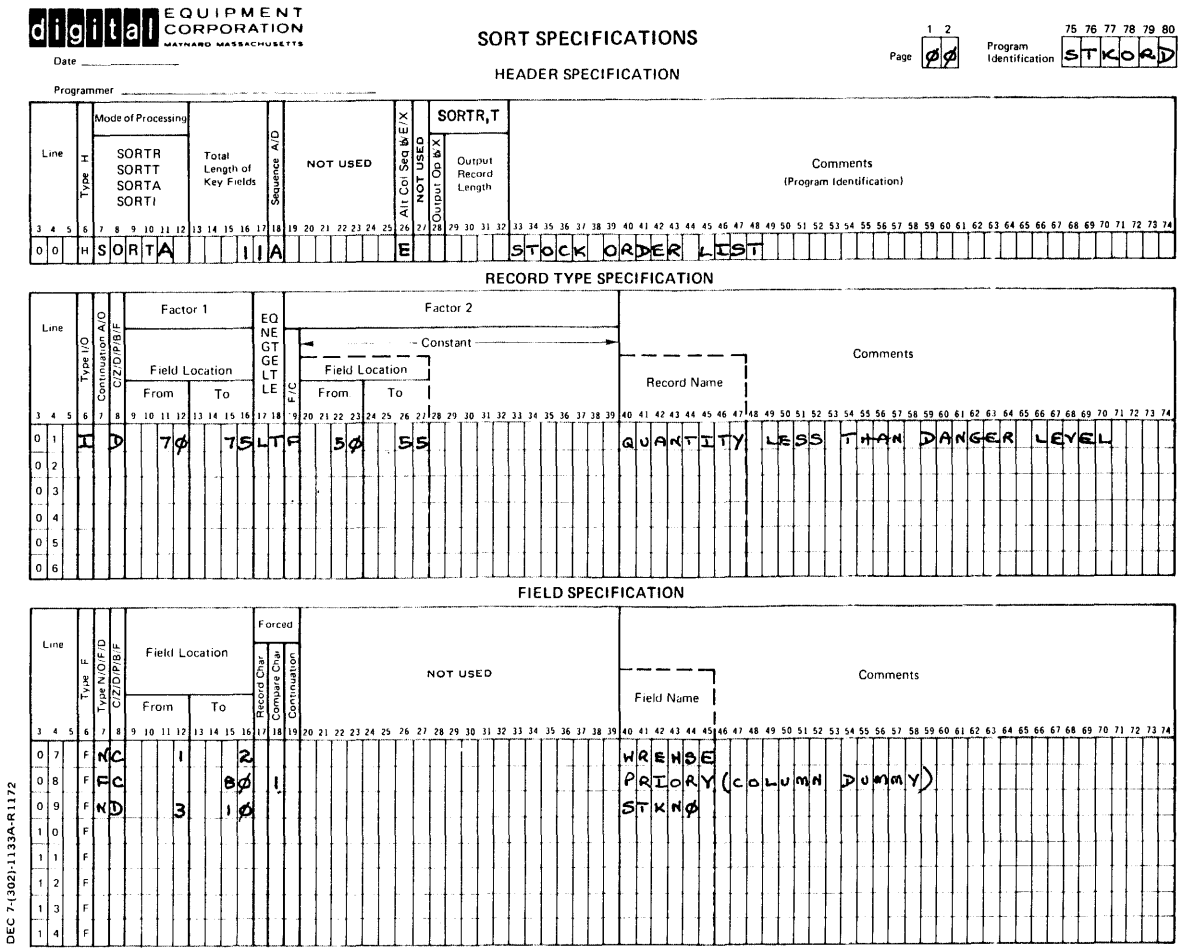
Warehouse 1 Stock Numbers of Items whose quantity is low

Stock Numbers of Items whose quantity is acceptable

Warehouse 2 Stock Numbers of Items whose quantity is low

Stock Numbers of Items whose quantity is acceptable

Warehouse 3 etc.




DEC 7-(302)-1133A-R1172

Figure C-5a Stock Order List, Page 1

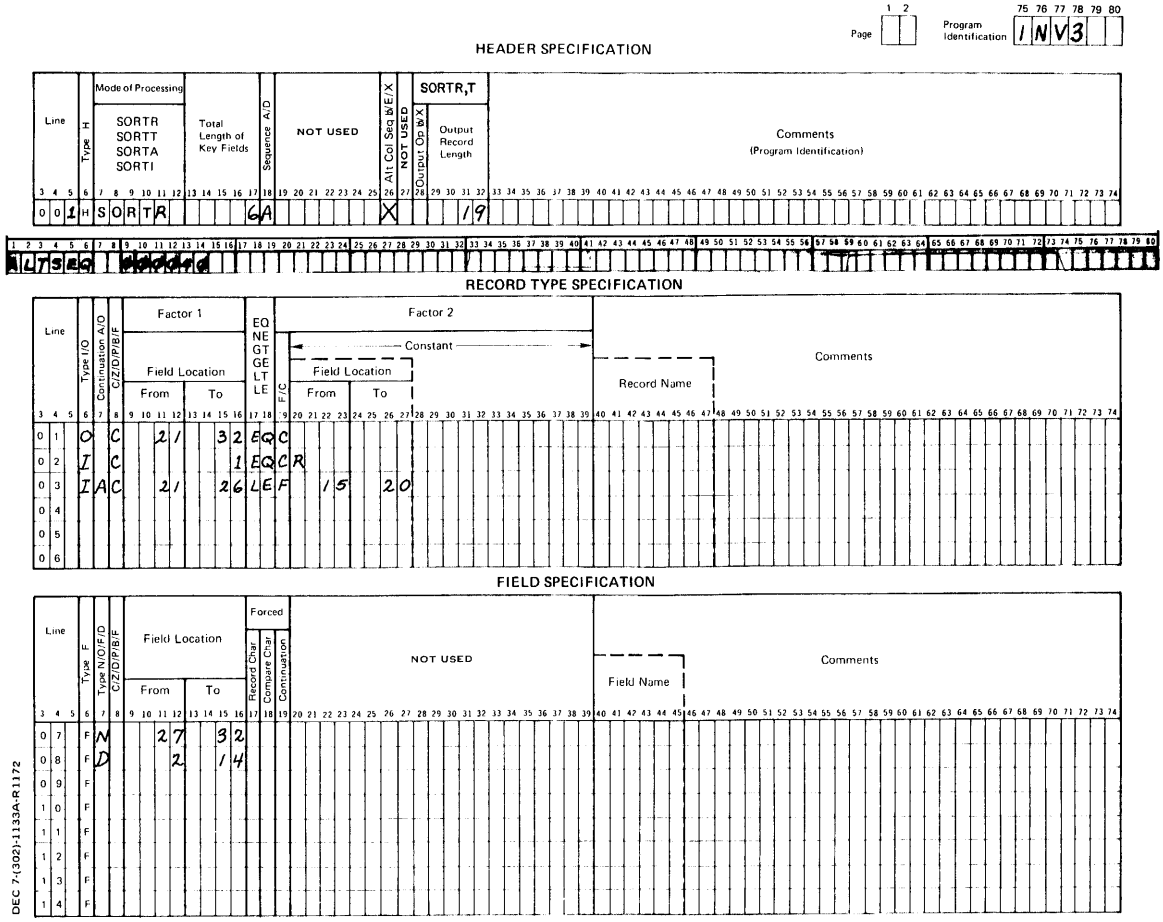



Figure C-5b Stock Order List, Page 2

APPENDIX D

# INTERNAL OPERATIONS

## D.1 PROCESS DESCRIPTION

The material in this Appendix is included for the benefit of the more experienced programmer. This information is not necessary for the successful use of SORT by the less experienced user.

The SORTS subroutines, which make up the bulk of the SORT program, consist of two phases: a sorting and distribution phase and a merge phase. During the sorting and distribution phase, the RELES subroutine accepts records from the input file and places them in an internal sort area called the sort tree. Each time the tree is filled, RELES distributes intermediate strings or runs of sorted records to the scratch files. The average length of these runs depends on the size of the record and its key. Also, the more nearly sorted the input file is, the longer the runs are. Because the algorithm of the SORT package is replacement selection, the average length of a run is twice the number of records that fit into the sort area.

At the end of the input file, the RELES routine finishes the sorting and distribution phase and RSORT calls the merge phase. The merge phase consists of two subphases: an intermediate merge, the MERGE subroutine, and a final merge, the RETRN subroutine. The MERGE subroutine reads the series of runs from the scratch files, merges them into smaller numbers of longer runs and writes them out to the scratch files until there is only one run per scratch file. The RETRN subroutine then merges the runs together to create a single stream of records to be returned to you one at a time.

## D.2 THE SORT WORK AREA

The entire work area of SORT is in the psect AAAAAA. Psect AAAAAA has an initial length of 0, which you can expand by using the EXTSCT command in the task build command file. The size of this psect controls the ultimate task image size. In general, the larger the AAAAAA psect is, the faster SORT runs.

SORT calculates the size of the work area by subtracting the address of the global location WA from that of the global location BOTTOM. These two global symbols mark the upper and the lower limits, respectively, of the SORT work area.

The SORT work area is used for several purposes, which are illustrated in Figure D-1. Figure D-1a shows the layout of the work area at the end of the command decoding phase. RMS uses space near the lower end of the work area for file control blocks and SORT uses space near the upper end for internal key information. The upper area is initially

set up to hold 10 keys. Its size is stored in the global location KYAREA in the segment SORTCX. To change the size of the key area, enter the following command in the task build command file:

```
GBLPAT=SORTCX:KYAREA:x
```

where:

x is the number of keys times 30 and is expressed in octal.

The pointers WORKHI and WORKLO contain the addresses of the upper and lower limits, respectively, of the remaining work area space and are maintained throughout the sorting process.

Figure D-1b shows the physical layout of the SORT work area before RSORT begins. An additional area from the low end of the available work area space is turned over to RMS for internal control blocks. SORTS also sets up the input file buffer. Note the changed positions of WORKHI and WORKLO. The positions have changed because the values of the addresses in these two locations have changed to reflect the new boundaries of available work area space.

If the input file is designated as sequential, SORT uses the RMS large disk blocking feature. Large disk blocking, also called big blocking, means that if you have enough core, you can call several blocks of your input file off the disk at once, instead of just one block at a time. Big blocking can be applied to all the files you use in your sort and, if you can use it, speeds up the sorting process by cutting down the number of times you have to do a disk seek.

The SORT control program initializes the work area for the actual sorting process as follows:

1. RSORT divides the available work area core space into quarters. One quarter, rounded down to the nearest 512-byte block, is used as an input block buffer.

2. The number of blocks in the input block buffer is passed as a parameter to RSORT to control its scratch file blocking factor. To override the scratch file blocking factor, specify an explicit blocking factor with the BUCKETSIZE switch on the input side.

3. The amount of space remaining in the work area after the space for the input block buffer is deducted is passed to RSORT as a parameter.

If the big blocking factor is 0, then half of the space left after the input block buffer is deducted is allocated for scratch file buffer space. If the big blocking factor is not 0, then that many 512-byte blocks are allocated for scratch file buffer space. (The big blocking factor is an octal number.) In either case, the leftover space is used to store the sort tree and the record areas. You can change the big blocking factor on the scratch files by using this command at task build time:

```
GBLPAT=SORTS:$RBGBF:x
```

where:

x is the big blocking factor in octal.

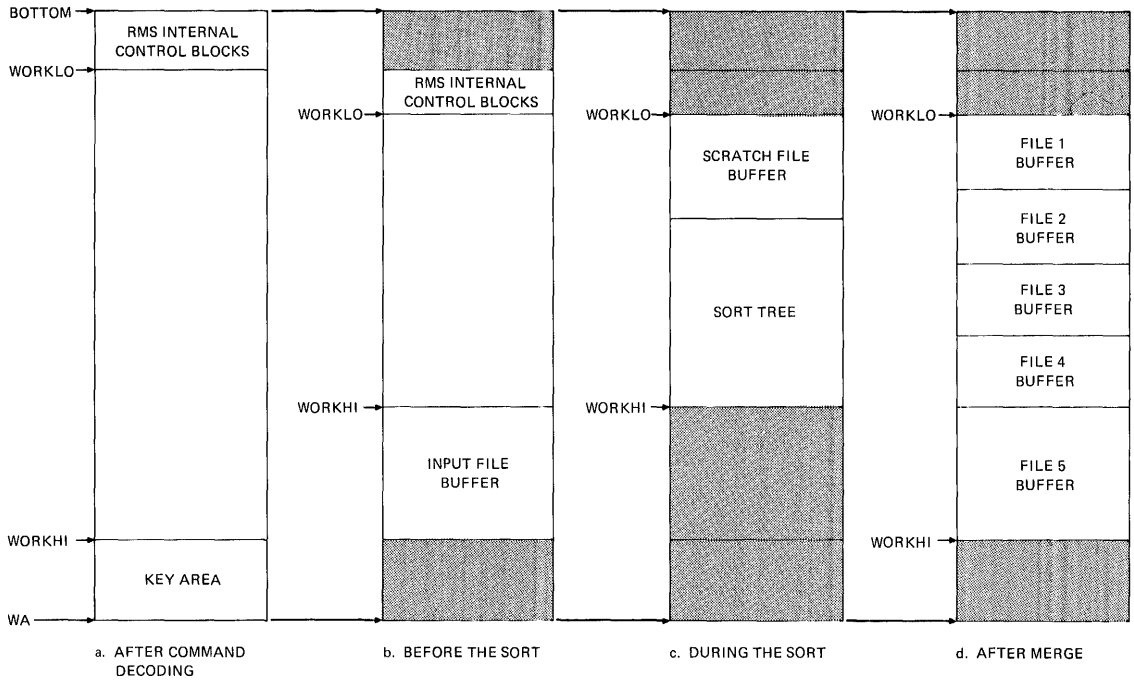



Figure D-1 SORT Work Area Allocation

The first record in the input file is read and the actual sorting process begins. SORTS handles the actual sorting and distribution of the input records. The physical core layout for the SORT work area resembles Figure D-1c.

At the end of the record distribution process, the input file buffer is no longer needed, so SORTS releases it. If the output file does not need more than two blocks of buffer space - that includes all sequential files and any relative files with bucketsizes less than three - the input buffer space is added back to the work area and WORKHI is adjusted accordingly.

At the start of the merge phase, SORTS divides the available work area space into as many buffers as there are scratch files specified. Figure D-1d shows the default - five buffers. One of the buffers is larger than the rest and the ratio between its size and the size of a small buffer is:

$$\sqrt{n-1} : 1$$

where:

n is the number of scratch files.

You can see that the more scratch files you specify, the larger the discrepancy in sizes between the large buffer and one of the small ones.

After the intermediate merge subphase and before the final merge, SORTS turns over the large buffer to RMS for use on the output file. Big blocking is enabled for sequential files, and the final merge takes place.

## D.3 PERFORMANCE PARAMETERS

Two main concepts you must consider when you try to increase the efficiency of SORT are:

- Determining the most efficient size of the work area
- Dividing the work area size among competing needs

The advantages of giving more space to the input and output buffers, at the expense of the sort tree are:

- Larger blocking factor
- Lower disk access and seek time

But a smaller sort tree results in:

- More intermediate runs
- A longer merge phase run

The default proportions for the input and output buffers and the sort tree are 25, 25, and 50 percent, respectively.

During the merge phase, RETRN uses almost all of the work area space for scratch file buffers. By assigning a large number of scratch files, you reduce the number of merge runs and general record shuffling. But if you lack the core space necessary for a large, fast-running sort, more scratch files can be a handicap. More scratch files in a small space means smaller files can be a handicap. More scratch files in a small space means smaller I/O buffers per file and therefore greater disk access and seek times.

Table D-1 shows you how to change seven factors that affect the speed and efficiency of SORT. The Result column reflects the effect of an increase in the factor being on the run time of the sort. Plus (+) means a faster sort; minus, a slower one; zero, no effect. The Change column shows the variables that you should change to affect the factor being considered. The Time and Place column legends translate as follows:

- TKB - make the change to the Task Builder command file at task build time.
- RSORT - make the change to the RSORT parameter list.
- IFS - make the change to the input file switch on the command file at task build time.

Table D-1
User-Accessible SORT Parameters

| SORT Parameter | Variable to Change | Result | Time and Place |
|---|---|---|---|
| Work Area Size | EXTCST=AAAAAA:n | + | TKB |
| | IWKSIZ | + | RSORT |
| Input Buffer Size | /BU:n | + | IFS |
| Scratch File Buffer Size | GBLPAT=SORTS:$RBGBF:n | + | TKB |
| | BIGBUF | + | RSORT |
| Key Area Size | GBLPAT=SORTCX:KYAREA:n | - | TKB |
| Number of Scratch Files | GBLPAT=SORT:FILES:n | 0[1] | TKB |
| Scratch File Clustersize | GBLPAT=SORTS:$RSCLV:n | + | TKB |
| | SCRLV | + | RSORT |
| | /SI:n | + | IFS |
| Contiguous Scratch Files | GBLPAT=SORTS:$RSCTG:1 | + | TKB |
| | /CO | + | IFS |
| Scratch File Preallocation | GBPAT=SORTS:$RPALQ | + | TKB |
| | PRIALQ | + | RSORT |
| | /AL:n | + | IFS |

[1]Detrimental if you lack sufficient core to run a large, fast sort.

NOTE

The /AL, /CO, and /SI switches apply only to the temporary scratch files when used on the input side. This usage is intended only for programmers who are completely familiar with the effects of these switches.

# APPENDIX E

# SORT INSTALLATION

## E.1 GENERAL COMMENTS

SORT is an independent task that can be run under control of RSTS/E (V06B-02), RSX-11M (V3.0), or IAS (V2.0). SORT is neither shareable nor reentrant, so multiple calls to SORT create multiple copies of SORT in core storage.

## E.2 SORT KIT CONTENTS

The SORT kit contains individual command files for you to use to build SORT on your operating system. These files are:

- For full RMS-11K (including indexed) support

  SRTRIM.CMD - RSTS/E with RMS11 Runtime System
  SRTEIM.CMD - RSTS/E with RSX Runtime System
  SRTMIM.CMD - RSX-11M and IAS

- For sequential and relative only RMS-11 support

  SRTRRM.CMD - RSTS/E with RMS11 Runtime System
  SRTERM.CMD - RSTS/E with RSX Runtime System
  SRTMRM.CMD - RSX-11 M and IAS

When you build SORT you must use one of these files, depending on which operating system and what kind of RMS-11 support you have.

The Task Builder command file contains a number of parameters that can be changed and options that can be included by the system manager to tune SORT to your installation's needs. These parameters control the size of the in-core work area (the scratch pad) which controls the task size, and the default logical unit and device assignments for the files SORT uses. Chapter 5 and Appendix D contain a full treatment of these parameters and other options. You can change the command file after it has been taken from the distribution medium and before the task build is started. To change just the task image size, see Section F.3.

As supplied by DIGITAL, the command file builds the largest possible task (28K words on RSTS/E, 32K words on RSX or IAS) on account number [1,2], the library account, under the name SORT.TSK. The program uses five scratch files as the default and finds and puts all files on the public disk structure.

### E.2.1 Building the SORT Task for RSX-11M or IAS

RSX-11M users can build and install SORT by using this procedure:

1. Use FILEX or PIP to transfer SORT from the distribution medium to a privileged account on public disk space:

```
>FLX SY:/RS=MM0:[1,2]*.*/DO
```

2. At this point make any changes to the Task Builder command file:

3. Build the task:

```
>TKB @SRTMIM        (for full RMS-11K
                    support on RSX-11M or IAS)
```

4. Install the task image:

```
>INS [1,2]SORT       (the default task
                      name is ...SRT)
```

5. Erase the files if no longer needed:

```
>PIP @SRTERS
```

6. Invoke SORT:

```
>SRT
SRT>  (SORT is now ready)
```

### E.2.2 Building the SORT Task for RSTS/E

NOTE

The following procedures for installing SORT-11 apply only to RSTS/E V6C and earlier versions. For instructions on SORT-11 installation, refer to the RSTS/E System Generation Manual.

RSTS/E users can build and install SORT by using this procedure:

1. Use PIP to transfer the file from the distribution medium to a privileged account on public disk space:

```
PIP *.*/CO=MM0:[1,2]*.*
READY
```

2. At this point make any changes to the Task Builder command file.

3. Build the task:

```
RUN $TKB.TSK
TKB> @SRTRIM         (full RMS-11 support)
TKB>//
READY
```

4. Make the task image executable by all users:

```
PIP $SORT.TSK<104>/RE
READY
```

5. Add the "SRT" CCL command:

```
RUN $UTILTY
UTILTY V06B-03 RSTS V06B-02 TS/1
#CCL SRT-=$SORT.TSK;0
#^Z
READY
```

6. Spool off and save the build map:

```
QUE SORT.MAP/DE
READY
```

7. Erase the files if no longer needed:

```
PIP @SRTERS
READY
```

## E.3 CHANGING THE SORT TASK IMAGE SIZE

The total size of the installed task image is controlled by the size of the psect AAAAAA, the scratch pad work area. To make the task smaller, cut down the size of the psect as follows:

1. Use an editor to access the proper command file.
2. Find the line "EXTCST=AAAAAA:XXXXXX".
3. Copy the number represented by XXXXXX here. A.
4. Determine the desired task size reduction in 4K word increments (e.g. 12KW = 3) B.
5. Multiply the number on line B by 20000(8). C.
6. Subtract line C from A using octal arithmetic. D.
7. Replace XXXXXX in the command file with the value on line D.
8. Continue with the build.

APPENDIX F

# RMS I/O STATUS CODES

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

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

Table F-1
RMS Error Status Codes

| Decimal Value | Description |
|---|---|
| -16 | Operation aborted: out-of-stack save area or in-core data structures corrupted. |
| -32 | Files-11 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 F-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. |
| -336 | Close function failed (RSTS/E operating system only). |
| -352 | Invalid COD field in XAB or this XAB type is illegal for the organization or operation. |
| -368 | Files-11 ACP could not create file. |
| -384 | No current record: operation not immediately preceded by a successful $GET or $FIND. |
| -400 | Files-11 ACP deaccess error during $CLOSE. |
| -416 | Invalid area number in DAN field of key definition XAB. |
| -432 | Record accessed by RFA access mode has been deleted. |
| -448 | 1. Syntax error in device name.<br>2. No such device.<br>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 F-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 | Files-11 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$BID. |
| -656 | 1. Record operation attempted was not declared in FAC field of FAB at open time.<br>2. Invalid contents in FAC field.<br>3. FB$PUT not present in FAC for $CREATE operation. |
| -672 | File already exists (attempted $CREATE operation). |
| -680 | Invalid file ID. |
| -688 | Invalid combination of values in FLG field of key definition XAB (e.g., no duplicates and keys can change). |
| -704 | File locked by another user -- your program cannot access the file because its sharing specification cannot be met. |
| -720 | Files-11 ACP Find function failed. |
| -736 | File not found. |
| -752 | Syntax error in file name. |
| -768 | Invalid file options. |
| -784 | Device full: cannot create or extend file. |
| -800 | Invalid area number in IAN field of key definition XAB. |
| -816 | Index not initialized (this code can only occur in the STV field when STS contains ER$RNF). |

(Continued on next page)

Table F-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:<br>1. Attempting a $TRUNCATE operation to a non-sequential file.<br>2. Attempting an $ERASE or $EXTEND operation to a magnetic tape file.<br>3. Issuing a block mode operation (e.g., $READ or $WRITE) to a stream not connected for block operations.<br>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:<br>1. As input to random $GET or $FIND operation, or<br>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 F-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 | Files-11 ACP could not mark file for deletion. |
| -1104 | 1. Maximum record number field contains a negative value during $CREATE of relative file.<br>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:<br><br>1. Record Format is fixed, or<br>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. |
| -1200 | Open function failed (RSTS/E operating system only). |
| -1216 | XABs in chain not in correct order:<br><br>1. Allocation or key definition XABs not in ascending (or densely ascending) order.<br>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 F-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$BID. Refer to Section A.3 of this Appendix. |
| -1328 | 1. Illegal values in record access mode (RAC) field of RAB.<br>2. Illogical value in RAC field (e.g., RB$KEY with a sequential file). |
| -1344 | 1. Illegal values in record attributes (RAT) field of FAB during $CREATE.<br>2. Illogical combination of attributes (e.g., FB$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.<br>2. Specified record format is illegal for file organization. |
| -1440 | Target bucket locked by another task or another stream in the same program. |
| -1456 | Files-11 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 F-1 (Cont.)
RMS Error Status Codes

| Decimal Value | Description |
|---|---|
| -1568 | Record size specified in RSZ of RAB during $PUT or $UPDATE is invalid:<br><br>1. RSZ equals 0.<br>2. RSZ exceeds maximum record size (MRS) specified when file created.<br>3. RSZ not equal to size of Current Record for $UPDATE operation to a sequential file on disk.<br>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:<br><br>1. UBF contains 0, or<br>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 entire set of 128 ASCII characters.

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. Used in FORTRAN IV representation.

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


/AL, 3-1
/BL, 3-1
/BU, 3-1
/CO, 3-3
/DE, 3-3
/FI, 3-3
/FO, 3-3
/IN, 3-3
/KE, 3-4
/PR, 3-4
/RE, 3-4
/SE, 3-4
/SI, 3-4

$SORT,
  $RUN, 2-1
  RUN, 2-2

Access from a random file,
  direct, 1-1
  sequential, 1-1
Address routing sort, 1-1, 1-8, 1-9, 1-10, 3-4
Allocation,
  file, 5-2
Allocation switch, 3-1
Allocation, figure,
  SORT work area, D-3
Alternate collating sequence, 1-6, 3-6, 3-7
Alternate key, 1-2
ALTSEQ, 1-6, 3-7
ALTSEQ format and notes, 3-10
ALTSEQ records, 3-6
ANSI format, 1-3
Ascending order, 1-4
ASCII, 1-10
ASCII character set, table, A-2
ASCII data, 1-2
ASCII zone, 3-4
Assignments table,
  LUN, 5-1
Available internal storage space, 1-8

Batch mode,
  running SORT in RSTS/E, 2-2
Binary data, 1-2, 1-9

Block,
  512-byte, 3-1
Blocking,
  big, D-2
Blocks per bucket, 3-1
Blocksize switch, 3-1
Boolean expression, 3-13
Bucket,
  blocks per, 3-1
Bucket size, 3-1
Bucketsize switch, 3-1
Buffer size ratio,
  scratch file, D-3
Byte,
  null, 1-10
  padding, 5-9

Calling sequence,
  SORT subroutine, 5-10
Card reader, 1-2
Cards, 1-3
Changing SORT defaults, 3-1
Changing sort order within a sequence, 3-7
Character set, table,
  ASCII, A-2
  EBCDIC, A-3
Characters, equal digits,
  printable, table, B-2
Characters, equal zones,
  printable, table, B-3
Cluster size,
  file, 3-5
Code,
  device, 2-3
  user identification, 2-3
Codes,
  RMS error, 4-7
  RMS error status, F-1
  SORT error, 5-5
Codes, table,
  device specification, 2-4
Collating sequence,
  alternate, 1-6, 3-6, 3-7
  normal, 1-6
Collation,
  nonstandard, 3-5
Command decoder errors, 4-1
Command file,
  indirect, 2-2
Command string, 1-1, 1-4, 2-2
  errors, 1-8
  format, 2-1, 2-2

Command string (Cont.),
  functions, 1-4
  SORT, 2-1
  switches, 2-2
Comments, 3-8
Conditional keys, 3-17
Conditions,
  error, 4-1
Configuration,
  system, 1-2
Console, 1-3
Contiguous switch, 3-3
Control fields, 1-1
Control program, 1-7
  errors, 4-6
Control-phase, 4-1
Controlling SORT, 1-1
Controlling SORT options, 3-1
Controls,
  sorting process, 3-5
Conventions,
  file naming, 5-1

Data,
  ASCII, 1-2
  binary, 1-2, 1-9
  EBCDIC, 1-2
  entry, size switch defaults, 3-5
  field, 1-2
    defining, 3-7
    locating, 3-7
    specifications, 3-7
  handling, 3-4
  irrelevant to SORT, 1-2
  key, 1-2
  output, 1-9
  relevant, 1-2
  restricted-format key, 3-7
Data base, 1-1
Decoder errors,
  command, 4-1
Defaults,
  changing SORT, 3-1
  file, footnote, 3-2
  file types, 2-3
  sort process, 1-8
  switch, table, 3-2
Defining a data field, 3-7
Defining a key, 3-7
Description,
  SORT process, D-1
Description sequence,
  field, 3-4
Device,
  code, 2-3
  file, input and output, 2-2
Device (Cont.),
  peripheral, 1-2
  specification codes, table, 2-4
  switch, 3-3
  types, table, 2-4
Different record formats, 1-7
Digits,
  printable characters, equal, B-2
Direct access from a random file, 1-1
Directing SORT, 1-5
Discussion,
  specification file example, 3-17
Disk, 1-2
  scratch files on, 1-8
  wasted space on, 5-2
Distribution process,
  record, D-3

EBCDIC,
  character set, table, A-3
  data, 1-2
ENDS, 5-9
Entry,
  common, of specification lines, 3-7
  field specification, 3-15
  forced key specification, 3-17
  header specification, 3-9
    notes, 3-10
  points, SORTS subroutines, 5-6
  record type specification, 3-17
  SORT specification, figure, 3-18
Error,
  codes,
    RMS, 4-7
    RMS status, F-1
    SORT, 5-5
  command decoder, 4-1
  command string, 1-8
  control program, 4-6
  I/O, 4-1, 4-5
  merge, 4-6
  messages, 4-1
  other, 4-6, 4-8
  presort, 4-6
  RSX-11M file primitive, 4-7
  RSX-11M attribute mismatch, 4-7
  SORT, 4-1

Error (Cont.),
  specification file, 1-8, 4-1, 4-3
  subroutine, 4-8
Example,
  SORT, C-1
  SORTR, 3-17
  specification file discussion, 3-17
Expression,
  Boolean, 3-13

Factor,
  big blocking, D-2
Field,
  control, 1-1
  data, 1-2
  defining a data, 3-7
  description sequence, 3-4
  first byte, key, 3-4
  interpretation, 1-2
  locating a data, 3-7
  key, 1-1, 1-5, 1-10, 3-4
  length, key, 3-4
  location, key, 1-5
  specification, 3-7, 3-14
    data, 3-7
    entries, 3-15
    key, 3-7
    notes, 3-16
    stripping key, 3-14
    table, 3-22
File,
  allocation, 5-2
  buffer size ratio, scratch, D-3
  cluster size, 3-5
  defaults, footnote, 3-2
  devices,
    input and output, 2-2
  errors,
    specification, 1-8, 4-1, 4-3
  example, discussion, specification, 3-17
  index, 1-1
  indexed, 1-2
  indirect command, 2-2
  information,
    RMS, 1-5
  input, 1-1, 1-9, 2-2, 3-17, 5-1
  input and output, 3-1
  internal work, 1-5
  merging scratch, 1-8
  moving records within a, 3-7
  name, 2-3
  naming conventions, 5-1
File (Cont.),
  needed, number of scratch, 1-8
  on disk, scratch, 1-8
  opening the input, 1-8
  optional specification, 1-1
  organization,
    input, 3-1
    RMS-valid, 1-2
  output, 1-8, 1-10, 1-11, 2-1, 2-2, 3-7, 5-1
  parameters, 1-5
    input and output, 2-2
  primitive errors,
    RSX-11M, 4-7
  random, direct access, 1-1
  relative, 1-2
  scratch, 5-1
  sequential, 1-2
  sequential access from a random, 1-1
  sorted, 2-1
  specification, 1-5, 1-8, 2-2, 2-3, 3-5, 3-14, 3-17, 4-3, 5-1
  structure, scratch, 5-3
  switch, 3-3
  type, 2-3
  types, default, 2-3
  uniform format input, 1-5
  usage, scratch, 5-3
First byte,
  key field, 3-4
FIXED format, 3-3
FIXED records, 1-8
Forced key specification entries, 3-17
Forced keys, 1-6
Form,
  figure, 3-8
  SORT specification, 3-7
Format,
  ALTSEQ, and notes, 3-10
  ANSI, 1-3
  command string, 2-1, 2-2
  FIXED, 3-3
  output, 1-7
  packed decimal, 3-4
  record, 1-5
    different, 1-7
  STREAM, 3-3
  switch, 3-3
  uniform input file, 1-5
  UNKNOWN, 3-3
  VARIABLE, 3-3
  variable record, 3-7
  variation, input, 1-7
FORTRAN IV number, 3-4
Functions,
  command string, 1-4

General data, irrelevant to
    SORT, 1-2

Handling,
  data, 3-4
Header specification, 3-6, 3-17
  entries, 3-9
  notes, 3-10
  table, 3-19
High-speed sorting, 1-8

I/O errors, 4-1, 4-5
IAS, 2-1
Identification by type,
  record, 1-7
Identification code,
  user, 2-3
Identifier,
  record, 1-2
Index,
  file, 1-1
  keys, 1-10
  record, 1-1, 1-10, 1-11
  sort, 1-1, 1-8, 1-10, 1-11, 3-4
Indexed file, 1-2
Indexed sequential switch, 3-3
Indices,
  record, 1-11
Indirect command file, 2-2
Information,
  key, 1-1
  prologue, 1-2
  RMS file, 1-5
  storage of sorted, 1-1
Initial sorting process, 1-8
Input and output files, 3-1
  devices, 2-2
  parameters, 2-2
Input file, 1-1, 1-9, 2-2, 3-17,
    5-1
  opening the, 1-8
  organization, 3-1
  uniform format, 1-5
Input format variation, 1-7
Input phase, 1-9
Installation,
  SORT, E-1
Interactive mode, 2-1
Internal operations,
  SORT, 5-1, D-1
Internal work files, 1-5
Interpretation,
  field, 1-2
Invoking SORT, 2-1
Irrelevant to SORT,
  general data, 1-2

Key,
  alternate, 1-2
  area size, D-2
  conditional, 3-17
  data, 1-2
    restricted-format, 3-7
  defining, 3-7
  fields, 1-1, 1-5, 1-10, 3-4
    first byte, 3-4
    length, 3-4
    location, 1-5
    specifications, 3-7
    stripping, 3-14
  forced, 1-6
    specification entries, 3-17
  in original form,
    record, 1-11
  index, 1-10
  information, 1-1
  locating a, 3-7
  primary, 1-2
  setup, 5-9
  significance,
    sort, 1-4
  switch, 3-4

Labels,
  magtape, 1-9
Length,
  key field, 3-4
Line number, 3-7
Lines,
  common entries of specifica-
    tion, 3-7
Locating a data field, 3-7
Locating a key, 3-7
Location,
  key field, 1-5
Logical Unit Number, 5-1
LUN assignments table, 5-1

Magtape, 1-2, 3-1
Magtape labels, 1-9
Management Services,
  Record, 1-1, 1-8
Maximum record size switch, 1-5
MERGE, 5-8
Merge errors, 4-6
Merge phase, 1-8
Merging scratch files, 1-8
Messages,
  SORT error, 4-1
Method,
  sorting, 1-1

Mismatch errors,
  RSX-11M, 4-7
Mode,
  interactive, 2-1
Moving records within a file, 3-7

Name,
  switch, 2-2
Naming conventions,
  file, 5-1
Nonstandard collation, 3-5
Normal collating sequence, 1-6
Notes,
  ALTSEQ format and, 3-10
  field specification, 3-16
  record type specifications, 3-13
Null byte, 1-10
Null records, 3-17
Number,
  FORTRAN IV, 3-4
  line, 3-7
  logical unit, 5-1
  page, 3-7
  project-programmer, 2-3
Number of scratch files needed, 1-8

Opening the input file, 1-8
Operations,
  SORT, 3-8
  SORT internal, 5-1, D-1
Optimization,
  sort, 5-2
Optional specification file, 1-1
Options,
  controlling SORT, 3-1
  sort processing, 1-8
  sorting process, 1-5
  sorting process, table, 1-9
Order, 1-4
  ascending, 1-4
  ascending sort, 3-4
  descending sort, 3-4
  general sort, 3-4
Organization,
  input file, 3-1
  SORT, 5-4
Organization, RMS-valid,
  file, 1-2
Other errors, 4-6, 4-8
Output data, 1-9
Output file, 1-8, 1-10, 1-11, 2-1, 2-2, 3-7
  devices, 2-2
  parameters, 2-2, 3-1, 5-1
Output format variation, 1-7
Output record, 1-11, 3-14

Package,
  SORTS subroutine, 1-7, 5-5
Packed decimal format, 3-4
Padding byte, 5-9
Page number, 3-7
Paper tape, 1-2
Parameters,
  file, 1-5
  input and output file, 2-2
  SORT, 3-4
  SORT performance, D-4
  SORT performance, table, D-5
Parts of SORT, 1-7
Performance parameters,
  SORT, D-4
  table, D-5
Peripheral device, 1-2
Phase,
  input, 1-9
  merge, 1-8
  SORT, 1-7
Pointer,
  record, 1-2
  relative record, 1-10
Points,
  SORTS subroutine entry, 5-6
Preallocation problems, 5-2
Presort errors, 4-6
Presort operation, 1-8
Primary key, 1-2
Printable characters, equal digits, table, B-2
Printable characters, equal zones, table, B-3
Printer, 1-3
Problems,
  preallocation, 5-2
Process,
  controls, sorting, 3-5
  default sort, 1-8
  description, SORT, D-1
  initial sorting, 1-8
  options, 1-5
    table, 1-9
  record distribution, D-3
  SORT, 1-8
  sorting, 1-5, 1-9
  switch, 3-4

Processing,
  options, sort, 1-8
  sort, 1-10
Program, control, 1-7
  errors, 4-6
Project-programmer number, 2-3
Prologue information, 1-2
Prompt,
  SORT, 2-2
  system, 2-1

Random file,
  direct access from a, 1-1
  sequential access from a, 1-1
Ratio,
  scratch file buffer size, D-3
Reel time, 2-1
Record Management Services, 1-1, 1-8
Records,
  ALTSEQ, 3-6
  distribution process, D-3
  FIXED, 1-8
  format,
    different, 1-7
    switch, 1-5
    variable, 3-7
  identification by type, 1-7
  identifier, 1-2
  index, 1-1, 1-10, 1-11
  keys in original form, 1-11
  moving, within a file, 3-7
  null, 3-17
  output, 1-11, 3-14
  pointer, 1-2
  selection, 1-5, 3-7
  size switch, maximum, 1-5
  sort, 1-1, 1-8, 3-4
  sorted, 2-1
  STREAM ASCII, 1-8
  strings of sorted, 1-8
  type specifications, 3-7
    entries, 3-11
    notes, 3-13
    table, 3-20
  VARIABLE length, 1-8
  within a file, moving, 3-7
Relative file, 1-2
Relative record pointers, 1-10
Relative switch, 3-4
RELES, 5-7
Relevant data, 1-2
Restricted-format key data, 3-7
Retrieval window size, 3-5
RETRN, 5-8

RMS, 1-1, 1-8, 3-1, D-2
  error codes, 4-7
  error status codes, F-1
  file information, 1-5
RMS-valid,
  file organization, 1-2
RSORT, 5-4, 5-6, 5-9
RSTS/E, 2-1, 2-2
  version number, omitted for, 2-3
RSX-11M, 2-1
  file primitive errors, 4-7
  mismatch errors, 4-7
$RUN $SORT, 2-2
RUN [1,2] SORT, 2-1
Running SORT, 2-1
  in RSTS/E batch mode, 2-2

Scratch files, 5-1
  buffer size ratio, D-3
  merging, 1-8
  needed, number of, 1-8
  on disk, 1-8
  structure, 5-3
  usage, 5-3
Selection,
  record, 1-5, 3-7
Sequence, 1-11
  alternate collating, 1-6, 3-6, 3-7
  changing sort order within a, 3-7
  field description, 3-4
  normal collating, 1-6
  SORT subroutine calling, 5-10
Sequential,
  access from a random file, 1-1
  file, 1-2
  switch, 3-4
Significance,
  sort key, 1-4
Size,
  bucket, 3-1
  file cluster, 3-5
  key area, D-2
  retrieval window, 3-5
Size ratio,
  scratch file buffer, D-3
Size switch, 3-4
  maximum record, 1-5
Size switch defaults, data entry, 3-5
$SORT,
  RUN, 2-1
  $RUN, 2-2

SORT, 1-1, 1-4, 1-8, 2-1, 3-17, 5-1
  address routing, 1-1, 1-8 to 1-10, 3-4
  command string, 2-1
  controlling, 1-1
  defaults, changing, 3-1
  directing, 1-5
  error codes, 5-5
  error messages, 4-1
  errors, 4-1
  examples, C-1
  general data, irrelevant to, 1-2
  Index, 1-1
  index, 1-8, 1-10, 1-11, 3-4
  installation, E-1
  internal operations, 5-1, D-1
  invoking, 2-1
  key significance, 1-4
  operations, 3-8
  optimization, 5-2
  options, controlling, 3-1
  order,
    ascending, 3-4
    descending, 3-4
    general, 3-4
    within a sequence, changing, 3-7
  organization, 5-4
  parameters, 3-4
  performance parameters, D-4
  parameters, performance, table, D-5
  parts of, 1-7
  phases, 1-7
  process, 1-8
    default, 1-8
    description, D-1
  processes, 1-8
  processing, 1-10
    options, 1-8
  prompt, 2-2
  Record, 1-1
  record, 1-8, 3-4
  running, 2-1
    in RSTS/E batch mode, 2-2
  specification,
    entries, figure, 3-18
    form, 3-7
      figure, 3-8
  subroutine calling sequence, 5-10
  switches, 3-1
  Tag, 1-1, 1-8, 1-9, 3-4
  work area, D-1
    allocation, figure, D-3
SORTA, 1-1, 1-9, 1-10
SORTC, 5-4
Sorted records, 2-1
Sorted file, 2-1
Sorted information,
  storage of, 1-1
Sorted records,
  strings of, 1-8
SORTI, 1-1, 1-10, 1-11, 3-17
Sorting,
  high-speed, 1-8
  method, 1-1
  process, 1-5, 1-9
    controls, 3-5
    initial, 1-8
    options, 1-5
      table, 1-9
SORTM, 5-4
SORTP, 5-4
SORTR, 1-1, 1-9, 3-14
  example, 3-17
SORTS, 5-4
  subroutine entry points, 5-6
  subroutine package, 1-7, 5-5
SORTT, 1-1, 1-9, 3-14
Space,
  available internal storage, 1-8
Space on disk,
  wasted, 5-2
Specification,
  device codes, table, 2-4
  entries,
    field, 3-15
      notes, 3-16
    forced key, 3-17
    header, 3-9
    notes, header, 3-10
    record type, 3-11
    SORT, figure, 3-18
  field, 3-7, 3-14
  file, 1-5, 1-8, 2-2, 2-3, 3-5, 3-14, 3-17, 4-3, 5-1
    errors, 1-8, 4-1, 4-3
    example, discussion, 3-17
    optional, 1-1
    switch, 1-5
  form, SORT, 3-7
    figure, 3-8
  header, 3-6, 3-17
  key field, 3-7
  lines, common entries of, 3-7
  record type, 3-7
    notes, 3-13
  table,
    field, 3-22
    header, 3-19
    record type, 3-20
  type, 3-7

SRT, 2-1
Status codes,
  RMS error, F-1
Storage of sorted information, 1-1
Storage space,
  available internal, 1-8
STREAM ASCII records, 1-8
STREAM format, 3-3
String,
  command, 1-1, 1-4, 2-2
  of sorted records, 1-8
  SORT command, 2-1
Stripping key fields, 3-14
Structure,
  scratch file, 5-3
Subroutine calling sequence,
  SORT, 5-10
Subroutine entry points,
  SORTS, 5-6
Subroutine errors, 4-8
Subroutine package,
  SORTS, 1-7, 5-5
Switch,
  allocation, 3-1
  blocksize, 3-1
  bucketsize, 3-1
  command string, 2-2
  contiguous, 3-3
  defaults, table, 3-2
  device, 3-3
  files, 3-3
  format, 3-3
  indexed sequential, 3-3
  key, 3-4
  maximum record size, 1-5
  name, 2-2
  process, 3-4
  record format, 1-5
  relative, 3-4
  sequential, 3-4
  size, 3-4
  SORT, 3-1
  specification, 1-5
System configuration, 1-2
System prompt, 2-1

Table, see also Table of Contents
  EBCDIC character set, A-3
  LUN assignments, 5-1
Tag sort, 1-1, 1-8, 1-9, 3-4
Terminal, 1-2
Terminators,
  valid, 4-2
Time,
  elapsed, 2-1
  real, 2-1
  wall clock, 2-1
Type,
  file, 2-3
  record identification by, 1-7
  specification, 3-7
Type specifications,
  record, 3-7
    entries, 3-11
    notes, 3-13
    table, 3-20
Types,
  default file, 2-3
Types, table,
  device, 2-4

UIC, 2-3
Uniform format input file, 1-5
Unit Number,
  Logical, 5-1
UNKNOWN file format, 3-3
Usage,
  scratch file, 5-3
User identification code, 2-3

Valid terminators, 4-2
Values,
  switch, 1-8, 3-1
VARIABLE format, 3-3, 3-7
VARIABLE length records, 1-8
Variation,
  input format, 1-7
  output format, 1-7
Version number, omitted for RSTS/E, 2-3

Wall clock time, 2-1
Wasted space on disk, 5-2
Window size,
  retrieval, 3-5
Work area,
  SORT, D-1
Work area allocation,
  SORT, figure, D-3
Work files,
  internal, 1-5
WORKHI, D-2
WORKLO, D-2

Zone,
  ASCII, 3-4
Zones,
  printable characters,
    equal, table, B-3