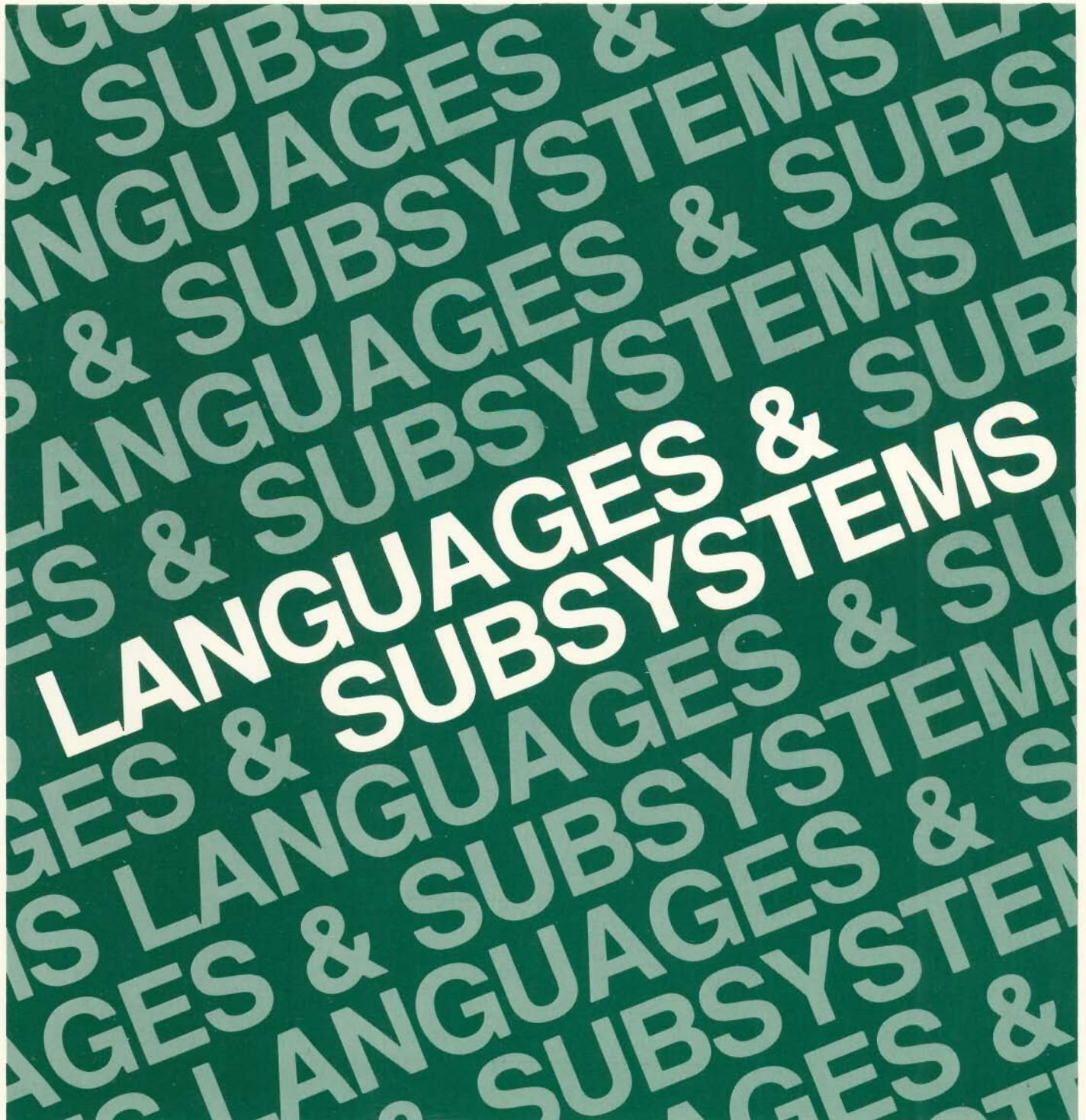


# V/R

RPG

utilities reference manual



**HP Computer Museum**  
**[www.hpmuseum.net](http://www.hpmuseum.net)**

**For research and education purposes only.**

# V/R

## HP 3000 Computer Systems



# RPG

## Utilities Reference Manual

Extra Function Sort for RPG/3000  
RPG Interactive System Environment/3000



19420 HOMESTEAD ROAD, CUPERTINO, CALIFORNIA 95014

### NOTICE

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

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

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

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

# LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

First Edition . . . . . Mar 1981  
Second Edition . . . . . Nov 1981

**\*\*\*NOTE\*\*\***

This manual corresponds with  
the software version of MPE V/R.

# PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover of the manual changes only when a new edition is published. When an edition is reprinted, all the prior updates to the edition are incorporated. No information is incorporated into a reprinting unless it appears as a prior update. The edition does not change.

The software product part number printed alongside the date indicates the version and update level of the software product at the time the manual edition or update was issued. Many product updates and fixes do not require manual changes, and conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition . . . . . Mar 1981  
Second Edition . . . . . Nov 1981

## PREFACE

This edition of the RPG UTILITIES MANUAL consists of two complete manuals for two independent RPG subsystems. The first is EXTRA FUNCTION SORT FOR RPG/3000. The second is RPG INTERACTIVE SYSTEM ENVIRONMENT/3000. To facilitate ease of use, each manual is complete in itself, with separate page numbers, Table of Contents, List of Illustrations, Appendices, and Index.

The two books are separated by a colored divider. Users may find it convenient to rearrange the contents, placing RISE/3000 first, if that is the subsystem most frequently used.





# PREFACE TO EXTRA FUNCTION SORT FOR RPG/3000 (XSORT)

This manual is intended for the use of inexperienced as well as seasoned programmers. To answer the needs of programmers with varied backgrounds, the manual presents information of varying complexity, especially in explaining the XSORT specifications. Most of this manual will be devoted to the five-part specifications. The information will be presented under the following headings:

## Overview of XSORT Specifications

shows you which columns of the specifications you must consider when you are preparing a sort. Overviews serve as a quick reminder to those who are familiar with the program.

## Overview of Column Entries

lists all possible entries for each column in the specifications. These are brief descriptions and explanations of the entries and their purposes.

## Column Entries

provide detailed discussions of the meanings of all possible entries, their relationships to other entries within the specification type and to other entries in other specification types. They are intended for new users of XSORT.

## Using XSORT Commands

gives you the various MPE commands necessary to use XSORT in job stream and interactive session, with different arrangements of input and output files.

## XSORT Applications

demonstrate how specifications are adapted to the requirements of particular jobs.

The more extensive the user's experience, the more readily will XSORT's versatility and efficiency become apparent. The manual takes into consideration that some users will be employing the program for the first time. Information and instruction are provided for several levels of experience so that the new user should be able to put XSORT to work after reading the manual.

Extra Function Sort for RPG/3000 is a utility of the HP 3000 and familiarity with the system will be of help to the user. The following manuals contain useful information for programmers who want to acquaint themselves with additional background materials:

Manual Title	Part Number
<i><u>RPG Reference Manual</u></i>	<u>32104-90001</u>
<i><u>Sort-Merge Reference Manual</u></i>	<u>32214-90001</u>
<i><u>MPE Commands Reference Manual</u></i>	<u>30000-90009</u>
<i><u>Console Operator's Guide</u></i>	<u>30000-90013</u>

In using XSORT, you may have programs and systems that interface with the Keyed Sequential Access Method (KSAM/3000), the IMAGE Data Base Management System (IMAGE/3000), and the Data Entry and Forms Management System (V/3000). For information about these systems, please read:

Manual Title	Part Number
<u>IMAGE/3000 Data Base Management Reference System</u>	<u>30000-90041</u>
<u>KSAM/3000 Reference Manual</u>	<u>30000-90079</u>
<u>V/3000 Reference Manual</u>	<u>32209-90001</u>

# CONTENTS

<b>SECTION I. INTRODUCTION</b>	<b>1-1</b>	<b>SECTION V. RECORD TYPE SPECIFICATIONS</b>	<b>5-1</b>
Characteristics of XSORT	1-1	Overview of Column Entries	5-2
How XSORT Works	1-2	Column Entries	5-3
Three Types of Sort	1-5	Columns 1-5 Line Number	5-3
Running the XSORT Program	1-6	Column 6 Spec Type	5-3
Supplying the Sort Specifications	1-6	Record Types	5-3
MPE Commands and Their Uses	1-9	Sections	5-3
Input Files	1-10	Sets	5-4
Work Space	1-10	Column 7 Continuation or Comments	5-5
Output File	1-11	Relating Columns 6 and 7 Entries — Include Sets	5-6
Data and Timing	1-11	Relating Columns 6 and 7 Entries — Omit Sets	5-7
Overview of XSORT Specifications	1-12	Mixing Sets	5-7
<b>SECTION II. HEADER SPECIFICATIONS</b>	<b>2-1</b>	Column 8 Data Type	5-8
Overview of Column Entries	2-2	Importance of the Column 8 Entry	5-8
Column Entries	2-3	Columns 9-16 Factor 1 Field Length	5-9
Columns 1-5 Line Number	2-3	Columns 17-18 Type of Comparison	5-10
Column 6 Line Type	2-3	Column 19 Field or Constant	5-11
Columns 7-12 Type of Sort	2-3	Columns 20-27 Factor 2 Field	5-11
Columns 13-17 Largest: Sum of Control Field		Columns 20-80 Factor 2 Constant	5-11
Lengths of any Record Section	2-3	Numeric Constants (Packed or Unpacked)	5-12
Column 18 Ascending or Descending Sequence	2-5	Unpacked Constants	5-12
Column 26 Collating Sequence	2-5	Packed Constants	5-13
Column 27 Print Option	2-5	Columns 40-80 Comments	5-13
Column 28 Output Option for Record-out Sort	2-6	<b>SECTION VI. FIELD DESCRIPTION</b>	<b>6-1</b>
Reasons for Dropping Control Fields	2-6	<b>SPECIFICATIONS</b>	
Columns 29-32 Output Record Length for		Overview of Column Entries	6-2
Record-out Sorts	2-6	Column Entries	6-3
Calculating Output Record Length Fields	2-6	Columns 1-5 Line Number	6-3
Columns 40-80 Job Description Comments	2-6	Column 6 Line Type	6-3
<b>SECTION III. ALTERNATE COLLATING SEQUENCE</b>	<b>3-1</b>	Column 7 Field Type or Comments	6-3
<b>SPECIFICATIONS</b>		Control Fields (N, O, or F) Entries	6-3
Rules for Coding ALTSEQ	3-2	Normal and Opposite Control Fields	
Specs Order Using ALTSEQ	3-3	(N and O) Entry	6-3
Programming with ALTSEQ	3-3	Opposite Control Fields	6-4
Effect of ALTSEQ Statements on Other Coding	3-3	Sequencing Information with Control Fields	6-4
Sample ALTSEQ Statements	3-4	Forced Control Fields	6-6
<b>SECTION IV. SPECIAL OPTION SPECIFICATIONS</b>	<b>4-1</b>	Conditional Force	6-6
Overview of Column Entries	4-2	Conditional Force on Normal or Opposite	
Column Entries	4-2	Control Fields	6-6
Columns 1-5 Line Number	4-2	Stand-alone Conditional Force	6-7
Column 6 S Entry	4-2	Force-All	6-7
Column 7 Address-out File First Record Number	4-2	Unconditional Force	6-8
Column 8 Inplace-sort Override	4-3	Data Fields (D) Entry	6-8
Columns 10-16 Maximum Number of Records		Comment Lines (*) Entry	6-8
to be Sorted	4-3	Column 8 C, P, U, or V	6-9
		Packed Control Fields (Normal or Opposite)	6-9
		Columns 9-16 Field Locations	6-10
		Field Length	6-11
		Column 17 Conditionally Forced Characters	6-11

# CONTENTS (continued)

Column 18 Forced Character . . . . .	6-12
Column 19 Continuation . . . . .	6-12
Summarizing Force Character Entries . . . . .	6-13
Examples Using Force Control Characters . . . . .	6-13
Example 1 Unconditionally Forced Character . . . . .	6-14
Example 2 Stand-alone Conditional Force . . . . .	6-16
Example 3 Conditional Force on Normal or Opposite Control Fields . . . . .	6-18
Example 4 Force-all . . . . .	6-20
Columns 40-80 Comments . . . . .	6-22

## SECTION VII. USING XSORT COMMANDS 7-1

Running XSORT . . . . .	7-2
About the Sample Sort . . . . .	7-3
Sort Specifications . . . . .	7-4
About the Job Stream Examples . . . . .	7-6
Example #1 – Job Stream with Specifications Imbedded; Single Input to Different Output . . . . .	7-6
Example #2 – Job Stream with XSORTTEXT; Single Input to Different Output . . . . .	7-9
Example #3 – Session with the Specifications Entered Interactively; Single Input to Different Output . . . . .	7-10
Example #4 – Session Using XSORTTEXT; Single Input to Different Output; Case 1 . . . . .	7-11
Example #5 – Session Using XSORTTEXT; Single Input to Different Output; Case 2 . . . . .	7-12
Example #6 – Job Stream with Specifications Imbedded; Single File, Inplace Sort . . . . .	7-13
Example #7 – Job Stream with Specifications Imbedded; Single File to Different Output that Already Exists . . . . .	7-14
Example #8 – Job Stream with Specifications Imbedded; Multiple Input File . . . . .	7-15
Example #9 – SORTC (Count-only Sort); Single Input, No Output . . . . .	7-16

## SECTION VIII. XSORT APPLICATIONS 8-1

Application #1 . . . . .	8-1
Application #2 . . . . .	8-2
Relating Record Types, Sections, and Sets – chart . . . . .	8-3
Application #3 . . . . .	8-6
Sample Sort 1 . . . . .	8-9
Sample Sort 2 . . . . .	8-13
Sample Sort 3 . . . . .	8-20

## APPENDIX A. COMPATIBILITY WITH OTHER SYSTEMS A-1

Compatibility with SORT3 . . . . .	A-1
Correlation of XSORT with SORT3 Formal Names . . . . .	A-2
Comparison of XSORT with IBM System/3 \$DSORT and IBM System 32 & 34 #GSORT . . . . .	A-3

## APPENDIX B. ERROR MESSAGES B-1

General Errors . . . . .	B-2
Syntax Messages . . . . .	B-2
Header Errors . . . . .	B-2
Alternate Collating Sequence Errors . . . . .	B-3
S Option Errors . . . . .	B-3
Record Type Errors . . . . .	B-3
Field Description Errors . . . . .	B-4
Unnumbered Errors . . . . .	B-4
Execution Messages . . . . .	B-5

## APPENDIX C. DATA INFORMATION C-1

Collating Sequence Table . . . . .	C-3
XSORT Alternate Collating Sequence Coding Worksheet . . . . .	C-9

## APPENDIX D. XSORT SPECIFICATION FORMS D-1

XSORT Specification Forms . . . . .	D-1
-------------------------------------	-----

INDEX . . . . .	I-1
-----------------	-----

**SECTION 1**

Figure 1-1. Uses of XSORT . . . . . 1-0  
Figure 1-2. Three Types of Sort . . . . . 1-4

**SECTION 2**

Figure 2-1. Calculating Control Field Length . . . . . 2-4

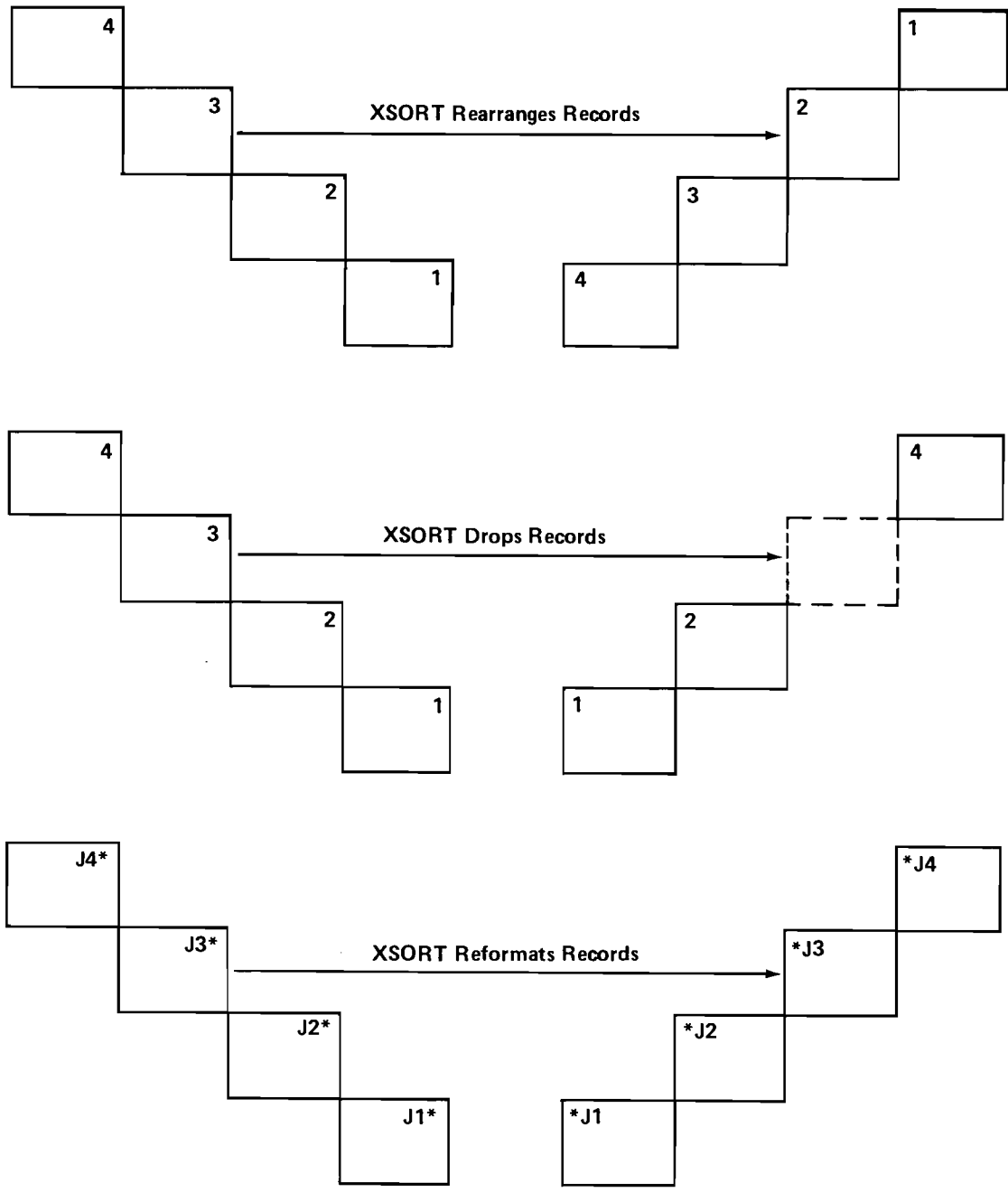


Figure 1-1. Uses of XSORT

Hewlett-Packard's Extra Function Sort for RPG/3000, XSORT, enables users to rearrange data in their files, drop records from their files, and reformat records, as shown in figure 1-1.

XSORT runs on an HP 3000 computer system and is primarily used in conjunction with the HP 3000 RPG programming language. Its origin was the unsupported utility SORT3 which was designed to perform most of the functions of IBM's System/3 \$DSORT and System/34 #GSORT. Similarities and differences are covered in Appendix A.

### **Characteristics of XSORT**

In extending and enhancing your sort capabilities, XSORT:

- uses from one to nine input files.
- employs multiple logical criteria to select records from MPE (disc and non-disc) and KSAM files, so you can process a subset of your input records.
- builds a different sort key for each type of record using input record fields and forced control fields. This facilitates the input of files with different data structures.
- reformats each type of record upon output to give you a choice of record layouts.
- can perform a Count-only pass on the input file(s). This permits you to conduct a preliminary search of your records, or provide statistics on selected subsets of records.

## Introduction

### How XSORT Works

XSORT follows a logical sequence in performing a sort. This sequence is described below:

XSORT reads a record from the input file.

XSORT checks your specifications to make sure the record is one you want to sort. You may not wish to sort all the records in the file you have specified.

XSORT builds a work record containing sort control fields and data fields. It also formats the data portion according to your specifications. Work record formatting is important because it controls the format of your output.

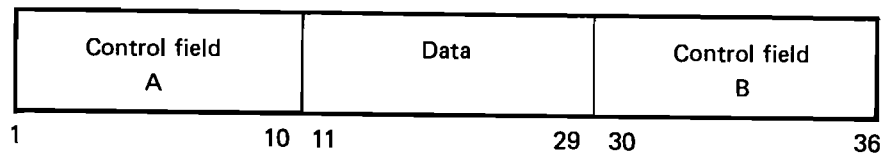
XSORT submits the work records to the SORT/3000 subsystem for sorting.

XSORT retrieves records from SORT/3000 after sorting is complete.

XSORT writes records into the output file, retaining or dropping control fields, according to your specifications.

#### NOTE

Control fields are input fields whose characteristics are used as a basis for the comparisons required by the sort. Control fields are discussed in detail in the section on Field Descriptions. (See Section 6.)

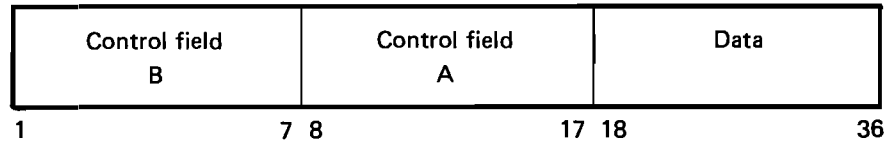


Suppose that you wanted to rearrange the input file above because a proposed sort job requires the following changes:

- a. the contents of positions 30 through 36 placed in positions 1 through 7 of the work record.
- b. the contents of positions 1 through 10 of the input file placed in positions 8 through 17 of the work file.
- c. the contents of positions 11 through 29 of the input record placed in positions 18 through 36 of the work record.



Here is the way XSORT would build your work record to reflect your specifications:



NOTE

The control fields must precede the data fields in the work record. They will be retained on output unless you wish to drop them. Reasons for retaining or dropping control fields will be discussed in the section on Header Specifications.



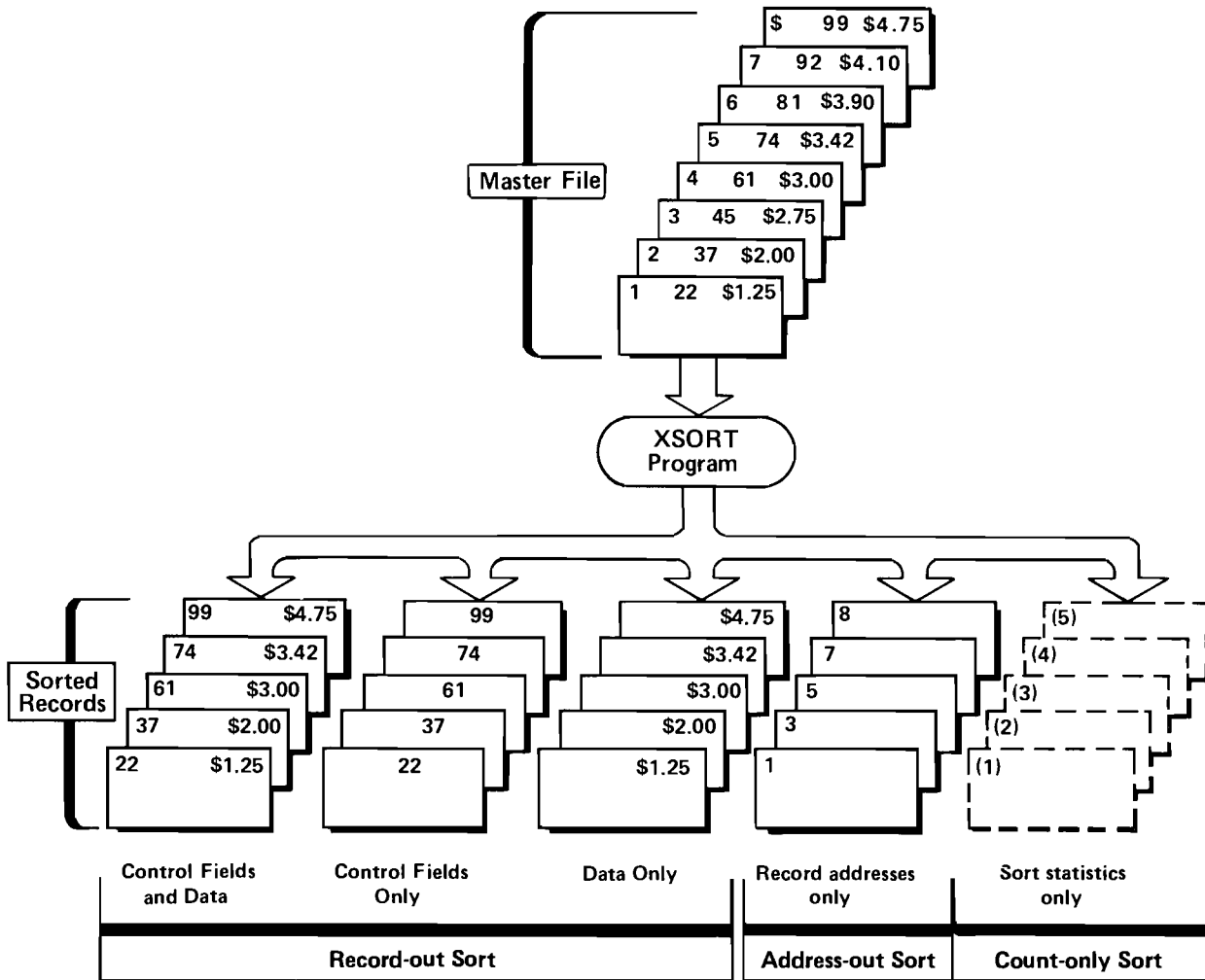


Figure 1-2. Three Types of Sort

### Three Types of Sort

XSORT offers three types of sort capability – Address-out, Record-out, and Count-only.

- Address-out sort (SORTA)

The output from an Address-out sort job consists of 4-byte binary relative record numbers of the records in the input file.

An Address-out sort allows you to process a file in sorted order without using up disc or tape space copying the entire file in order to process it.

- Record-out sort (SORTR)

The output from a Record-out sort is a file of complete records that are immediately available for processing. These sorted records can contain:

- control fields and data
- control fields only
- data only

A Record-out sort offers faster processing than the Address-out as well as enabling you to reformat files and to create new files from reformatted records.

- Count-only sort (SORTC)

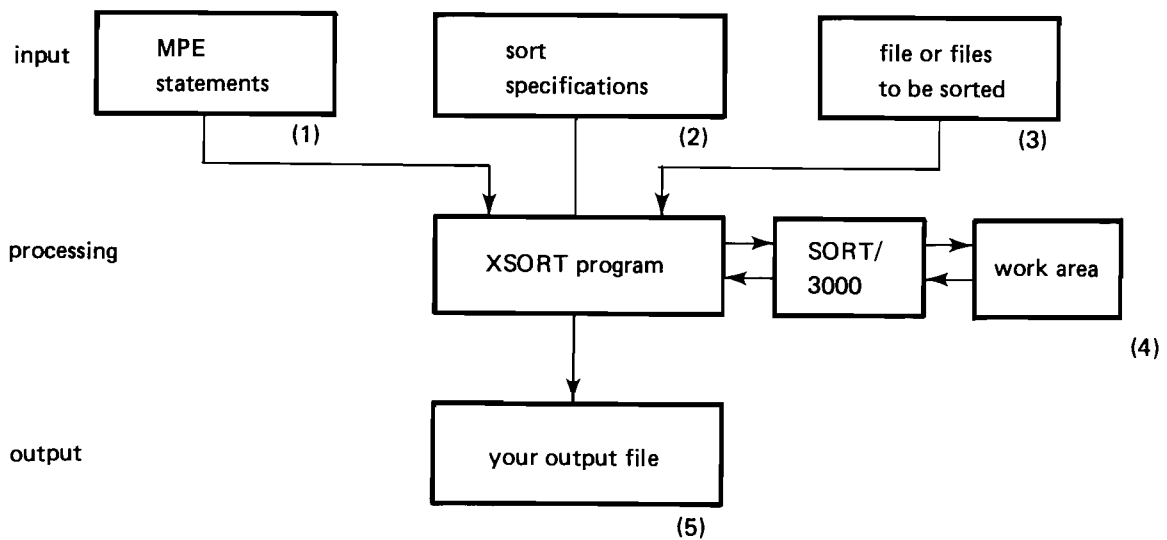
This sort performs a Count-only pass on the input file(s).

Using a Count-only sort, XSORT can provide you with a statistical analysis of the sort without actually performing the sort or producing an output file.

## Introduction

### Running the XSORT Program

XSORT sorts your files in this way:



1. MPE statements are your job or session control commands to the HP 3000.
2. Sort specifications include type of sort, sequence, record information and instructions for formatting output.
3. File or files can be KSAM or sequential MPE (including non-disc files such as magnetic tape).
4. XSORT passes work records to be sorted to SORT/3000 which, in turn, sorts them in a work file. Whatever work is required can then be performed without any disturbance to the input file.
5. Output files can contain:
  - the relative record numbers of the records in the input file
  - part or all of the records in the input file(s)
  - a count of the number of records selected for processing from the input file(s)

### Supplying the Sort Specifications

XSORT specifications are normally entered through a job stream. Interactive entry of specs is not recommended because XSORT does not prompt for input. There are two means of supplying sort statements: You may use either XSORTTEXT or \$STDIN.

XSORT reads your specifications from the file whose formal file designation is XSORTTEXT. By default, XSORTTEXT is associated with the actual file designator, \$STDIN. Specifications are read from XSORTTEXT (or \$STDIN) and any errors in syntax are detected and communicated to you on \$STDLIST.

XSORT specifications must always be entered in the following order.

1. Header
2. Alternate Collating Sequence
3. S Option
4. Record Type
5. Field Description



XSORT SPECIFICATIONS

Page \_\_\_\_ of \_\_\_\_

**HEADER**

Line Number	Spec. Type (H)	Type of Sort		Largest sum of control field lengths of any record section (in bytes)	Sequence (A or D)	g. (S)	j	Altern. Print L	Output record	Comments (any characters)																																																													
		SORTA, SORTR, or SORTC																																																																					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

Header specifications specify the type of sort job you are running, as well as other general control information.

Entries can extend to 80 columns →

← Insert ALTSEQ specifications here (if used).  
 ← Insert S OPTION specification here (if used).

**ALTERNATE COLLATING SEQUENCE**

A	L	T	S	E	O	Blank	Hexadecimal equivalents of paired characters		Entries can extend to 80 columns																																																														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
ALTSEQ entries specify an alternate collating sequence you wish to use.																																																																							

\* \* (Asterisks in columns 1 and 2 after last ALTSEQ line)

**S OPTION**

Line Number	Spec. Type (S)	D or T	N or Y	Reserved	Maximum number of records to be sorted	Comments (any characters)																																																																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
S Option specifications effectively offer additional header-type entries.						Entries can extend to 80 columns →																																																																	

**RECORD TYPE**

Line Number	Spec. Type (I or O)	Continuation (A, O, or *)	Factor 1		Rel.	Factor 2 (Field or Constant)		Record Name	Comments (any characters)																																																														
			Location			Constant (any ASCII characters)																																																																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

Record type specifications specify which records you are sorting.

Entries can extend to 80 columns →

**FIELD DESCRIPTION**

Line Number	Spec. Type (F)	Type (NO, F, D, or *)	P, U, E, or V	Location		Record character	Substitute character	Continuation	Forced	Reserved	Field Name	Comments (any characters)																																																											
				from	to																																																																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

Field description specifications specify how you want to sort the records.

Entries can extend to 80 columns →

## Introduction

You do not always need to use all the specification types for every sort; but the order in which the specifications are used doesn't vary. You always fill out Header Specifications first.

The Alternate Collating Sequence (ALTSEQ) and the S (Special) Option specifications can be ignored if they are not to be used in the sort. However, S Option specifications must be supplied when your input is from non-disc devices or when you are sorting multiple files.

It is not necessary to fill out the Record Type entries if you wish to have all the records in the file sorted and they are all written in the same format.

Number of Records to be Sorted	Format of Records to be Sorted	Order of Specifications																					
All the records in the file	All the same	1. Header** 2. Field descriptions																					
Some of the records in the file	All the same	1. Header** 2. Record Type specs 3. Field descriptions																					
All or some of the records in the file	Several different formats (record sections)*	<table style="border: none;"> <tr> <td style="border: none;">1. Header**</td> <td style="border: none;">}</td> <td style="border: none;">for first record format</td> </tr> <tr> <td style="border: none;">2. Record Type specs</td> <td style="border: none;">}</td> <td style="border: none;">for second record format</td> </tr> <tr> <td style="border: none;">3. Field descriptions</td> <td style="border: none;">}</td> <td style="border: none;">for second record format</td> </tr> <tr> <td style="border: none;">4. Record Type specs</td> <td style="border: none;">}</td> <td style="border: none;">for second record format</td> </tr> <tr> <td style="border: none;">5. Field descriptions</td> <td style="border: none;">}</td> <td style="border: none;">for second record format</td> </tr> <tr> <td style="border: none;">6. Record Type specs</td> <td style="border: none;">}</td> <td style="border: none;">One set for each additional record format.</td> </tr> <tr> <td style="border: none;">7. Field descriptions</td> <td style="border: none;">}</td> <td style="border: none;">One set for each additional record format.</td> </tr> </table>	1. Header**	}	for first record format	2. Record Type specs	}	for second record format	3. Field descriptions	}	for second record format	4. Record Type specs	}	for second record format	5. Field descriptions	}	for second record format	6. Record Type specs	}	One set for each additional record format.	7. Field descriptions	}	One set for each additional record format.
1. Header**	}	for first record format																					
2. Record Type specs	}	for second record format																					
3. Field descriptions	}	for second record format																					
4. Record Type specs	}	for second record format																					
5. Field descriptions	}	for second record format																					
6. Record Type specs	}	One set for each additional record format.																					
7. Field descriptions	}	One set for each additional record format.																					
<p>*This does not mean that the records in the input file must be grouped in sections by format type. Just the specifications must be grouped (one section of Record Type and Field Description lines for each format type). The input records can be in any order.</p> <p>** ALTSEQ and S Option, if used, would be entered following the Header Specification.</p>																							

Forms are provided in Appendix D to facilitate your organizing specifications before entering them. The forms include 72 column spaces but entries can be extended to 80 columns. Copies of the forms may be made at your convenience.

Detailed instructions on how to enter the specifications and the optional entries appear in Sections 2 through 6 of the manual.

## MPE Commands and Their Uses

You must supply the necessary commands to your system in order for it to perform the sort job you require. Instructions concerning files and file locations, and the name of the program are typical of the information supplied by MPE commands.

An unsupported program similar to XSORT called SORT3 has been in existence for some time. A discussion of the commands for SORT3 as compared to XSORT appears in Appendix A.

### Your MPE Commands Provide:

- Name of the program — XSORT.PUB.SYS
- Name of the file(s) you want to sort. The formal designator for a single file sort is XSORTIN. For multiple-file sorts formal file designators are XSORTIN1 through XSORTIN9. The multiple-file designators do not have to be continuous. You can skip file numbers, selecting XSORTIN1, XSORTIN3, and XSORTIN7, for example. However, you must start with XSORTIN1.
- Name of the file containing specifications of the sort. The formal file designator is XSORTTEXT, which defaults to \$STDIN.

XSORT verifies that all fields defined by the specifications are contained within the record size of the first file. Thus, it is up to the user to ensure that the specifications make sense for the records contained in the second through last input files of a multiple sort.

## Introduction

### INPUT FILES

XSORT can input sequential MPE files, including files on non-disc devices such as tape. (Please see Section 7 for the commands required.)

KSAM files may be input. Caution should be taken, however, because XSORT will skip deleted records. A deleted record has its first two bytes set to -1 (to hex "FFFF"). For Address-out sorts (SORTA), XSORT reads KSAM files in chronological sequence, and determines record numbers accordingly.

XSORT will normally assume that the first record for a KSAM file is record number 0. If you wish to start the Address-out numbering at 1, then you must supply the S Option spec with a "1" in column 7 and/or build the KSAM file with the Firstrec=1 option. (See Section 4 for further information regarding S Option specification.)

#### NOTE

XSORT will not process IMAGE data base files.

1. When you use more than input one file:
  - you cannot do an Address-out (SORTA) sort.
  - you cannot do an inplace sort, in which output file overlays input file.
  - XSORT bases its operations on the characteristics of the first file opened. All the other files are assumed to have the same, or compatible, characteristics (file type and record length).
  - you must be sure that all your specifications apply to all the records in all the files you wish to use in a multifile sort.
2. If single file input is from \$STDIN, the specification records cannot be from \$STDIN. They must be from a file referenced by the formal designator, XSORTTEXT.

### WORK SPACE

XSORT tells SORT/3000 how much work space will be required for performing the actual sort. For single-file input from disc, it determines the space needed to sort by the number of records in the input file. You must specify the maximum numbers of records to be sorted, using the S Option (See Section 4) in the following two cases:

- single-file non-disc input — including \$STDIN
- multiple-file input

If only a small percentage of the records from a single file sort are being selected and sorted, XSORT will make more efficient use of resources if a reasonably accurate number of records to be sorted is specified using the S Option.



## OUTPUT FILE

1. The output file is your sorted file. It can be a sequential MPE file or an already existing KSAM file. The output can be given in one of three formats:
  - Address-out
  - Record-out
  - Count-only (no output file is produced. XSORT only prints sort statistics).
2. The formal file designator is XSORTOUT.
3. If XSORTOUT does not already exist (non-disc files by definition already exist) then XSORT will create it as a job temporary disc file. If XSORTOUT does already exist, then it must be the same file as XSORTIN. In such an instance, the sort is an inplace sort. (This requirement can be overridden by coding a "Y" in column 8 of the S Option specs to allow output to an existing file other than XSORTIN.) XSORTOUT can never be the same as any of the XSORTIN1 through XSORTIN9 files in a multiple file sort.

## DATA AND TIMING

### Data Considerations

Maximum data input record size is 1000 bytes.

The collating sequence may be altered by the ALTSEQ specs. (See Appendix C for further information concerning data.)

### Timing Factors

The time it takes to run a sort depends on the following factors:

- the number of records to be sorted. The fewer the records specified for sorting, the shorter the job will be.
- record size. Larger records take longer.
- number of specifications you enter. The more specifications you have, the longer the sort will take.
- alternating sequence. If you use ALTSEQ, your program will take longer to run.

OVERVIEW OF XSORT SEQUENCE SPECIFICATIONS

Specifications for Record-Out Sort (SORTR)

The columns that are shaded must be considered when you are planning a Record-out sort job.

**XSORT SPECIFICATIONS HEADER**

Line Number		Spec. Type (H)	Type of Sort	Largest sum of control field lengths of any record section (in bytes)	Sequence (A or D)	Reserved	Alternate Coll. Seq. (S)	Print Option	Output Option (X)	Length of output record	Reserved	Comment																																										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
		H	SORTA, SORTR or SORTC																																																			
												Overall control information																																										

← Insert ALTSEQ specifications here (if used).  
 ← Insert S OPTION specification here (if used).

**RECORD TYPE**

Line Number	Spec. Type (I or O) Continuation (A, O, or *) C, P, or U	Factor 1	Rel.	Factor 2 (Field or Constant)	Comment																																																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
		Location from to		EQ, NE, LT, GT, LE, GE, F or C	Constant (any ASCII characters)		Location from to		Record Name																																													
		FIELD		C	CONSTANT		FIELD		Comparison of an input record field and a constant																																													
		FIELD		F	FIELD		FIELD		Comparison of two input record fields																																													

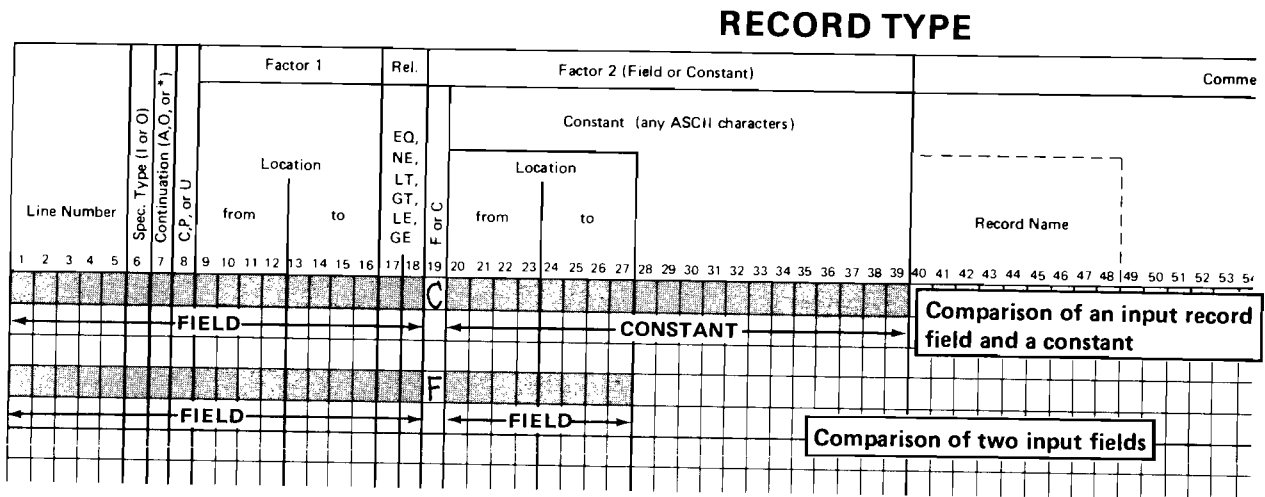
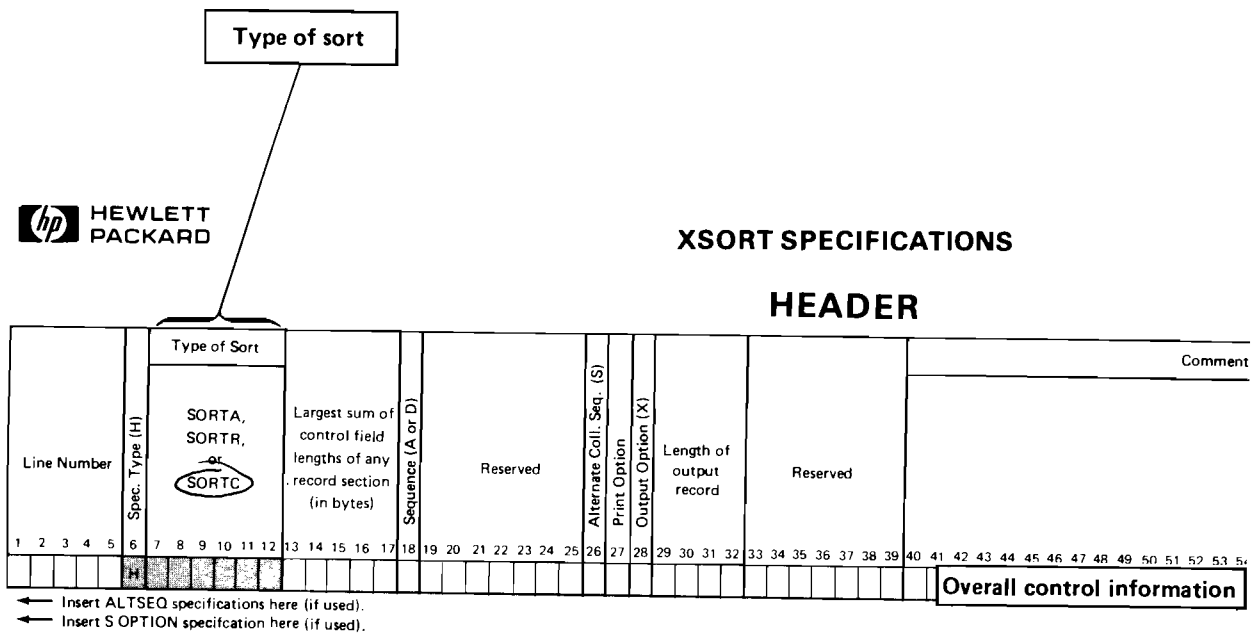
**FIELD DESCRIPTION**

Line Number	Spec. Type (F) Type (N, O, F, D, or *) P, U, C, or V	Location	Forced	Reserved	Field Name	Comment																																																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
		Location from to		Record character Substitute character Continuation	Field Name																																																	
		FIELD			Definition of normal control fields																																																	
		FIELD			Definition of opposite control fields																																																	
		FIELD			Definition of forced control fields																																																	



Specifications for Count-Only Sort (SORTC)

The Columns that are shaded must be considered when you are planning a Count-only sort.



When you enter Header Specifications you are telling XSORT

- the type of sort you want to do – Address-out, Record-out, or Count-only
- the format you have selected for the sorted file
- the information you wish to have printed to help you check for errors

Only one Header Specification is needed for each sort job.

## OVERVIEW OF COLUMN ENTRIES

COLUMNS	ENTRIES	EXPLANATION
1-5	Any value	Line number
6*	H	Header line identification
7-12*	SORTA	Address-out sort
	SORTR	Record-out sort
	SORTC	Count-only sort
13-17**	1-256	Largest sum of control field lengths for any Record Section (in bytes)
18**	A	Records in sorted file to be in ascending order by control fields
	D	Records in sorted file to be in descending order by control fields
19-25	Blank	Reserved
26	Blank	Use standard ASCII collating sequence in compare operations
	S	Use an alternate collating sequence in compare operations. ALTSEQ specs will define the collating sequence desired
27	0 or blank	Print: XSORT specifications Diagnostic messages Statistics Abort messages Display messages
	1	Print: Statistics Abort messages Display messages
	2	Print: Abort messages Display messages
	3	Print only display messages
28***	Blank	Retain control fields on output records in Record-out sort jobs
	X	Drop control fields from output records in Record-out sort jobs
29-32***	1-1000	Length of output records in Record-out sorts
33-39	Blank	Reserved
40-80	Any Characters	Comments

\* Columns that must be filled in for all sorts.

\*\* Columns that must be filled in for Address-out and Record-out sorts

\*\*\*Columns that must be filled for Record-out sorts.

## COLUMN ENTRIES

**Columns 1-5**      **line number**

Line number is an optional entry. XSORT does not check your columns 1-5 entries for sequence. Your entry here is intended as a convenience, a record you may use for future reference.

**Column 6**      **line type**

An entry of H identifies this as the Header line.

**Columns 7-12**      **type of sort**

In columns 7-12 you tell the program what type of sort you want carried out. SORTA means an Address-out sort. SORTR means Record-out sort. SORTC means a Count-only sort on the input file(s).

**Columns 13-17**      **largest sum of control field lengths of any Record Section (in bytes)**

The entry requires the following calculations:

- a. Compute the lengths of all control fields on Field Description specs (when column 7 is N,O, or F – see table below). Sum these totals separately for each record section (see definition of "Sections" on page 5-3).
- b. Place the largest of these sums in columns 13-17. The entry cannot exceed 256.

Additional information concerning this entry can be found under column 7 of the Field Description Specifications. Also, see figure 2-1, in which the largest control field sum of two record sections is calculated and the total entered in columns 13-17.

In computing the total lengths of control fields and work records the following guidelines should be noted:

	<b>If Col. 7 entry is</b>	<b>Add this to total length</b>
Control fields	N or O	Field length*
	F	1 – if Col. 19 (continuation) is blank, or 0 – if Col. 19 (continuation) is non-blank
Data fields	D	Field length* – if Col. 8 is blank, or
		1 – if Col. 8 contains V
* Actual length of the field as defined by its input record "location" (columns 9-16).		



### XSORT SPECIFICATIONS

#### HEADER

Line Number	Spec. Type (H)	Type of Sort	Largest sum of control field lengths of any record section (in bytes)	Reserved	Alternate Coll. Seq. (S)	Print Option	Length of output record	Reserved	Comments (any characters)
1-4	H	SORTA, SORTR, or SORTC							
5-11									Entries can extend to 80 columns

← Insert ALTSECO specifications here (if used).

← Insert S-OPTION specification here (if used).

#### RECORD TYPE

Line Number	Spec. Type (I or O) Type (N,O,F,D, or *) C, P, or U	Factor 1 Location from to	Rel.	Factor 2 (Field or Constant) Constant (any ASCII characters) Location from to	Comments (any characters)
1-5					
6-10	I		EQ, NE, LT, GT, LE, GE		
11-15	C	72	C		
16-18	O	72	O		

#### FIELD DESCRIPTION

Line Number	Spec. Type (F) Type (N,O,F,D, or *) P, U, C, or V	Location from to	Record character Substitute character Continuation	Reserved	Comments (any characters)
1-4					
5-10	F	4	8		
11-15	F	15	18		
16-18	F	10	14		
19-21	F	1	92		

#### RECORD TYPE

Line Number	Spec. Type (I or O) Type (N,O,F,D, or *) C, P, or U	Factor 1 Location from to	Rel.	Factor 2 (Field or Constant) Constant (any ASCII characters) Location from to	Comments (any characters)
1-5					
6-10	I		EQ, NE, LT, GT, LE, GE		
11-15					

#### FIELD DESCRIPTION

Line Number	Spec. Type (F) Type (N,O,F,D, or *) P, U, C, or V	Location from to	Record character Substitute character Continuation	Reserved	Comments (any characters)
1-4					
5-10	F	1	8		
11-15	F	15	18		
16-18	F	1	92		

Figure 2-1. Calculating control field length



**Column 18**      **ascending or descending sequence**

This entry indicates the sequence in which you want the records sorted.

Column 18 Entry	Sequence
A	Ascending sequence by control field
D	Descending sequence by control field

Sequencing will be done according to the ASCII collating sequence unless it is altered using ALTSEQ specifications.

**Column 26**      **collating sequence**

This entry determines the collating sequence you want XSORT to use in its comparing operations. Characters will be compared to ascertain if one is greater than, equal to, or less than another.

If you leave the column blank, XSORT will use the standard ASCII collating sequence.

An S in column 26 means you want to change the standard collating sequence. To do this you must include a set of ALTSEQ specifications after the Header specification. (See Section 3.)

If you specify an alternate collating sequence, you cannot use packed data for record selection factors or sort control fields. In other words, you cannot enter a P in column 8 of your Record Type specifications or in column 8 of your Field Description specs used to define control fields.

**Column 27**      **print option**

XSORT can print:

- XSORT specifications
- Diagnostic messages – for syntax errors in the specifications
- Statistics of the sort – record counts and timing data
- Abort messages – a “sortstone” identifying the primary cause of the failure
- Display messages – MPE abort messages and unnumbered diagnostic messages

Column 27 entries indicate which of the print options you want to have printed during the job:

Col. 27 Entry	Information Printed
0 or blank	XSORT specifications Diagnostic messages Statistics Abort messages Display messages
1	Statistics Abort messages Display messages
2	Abort messages Display messages
3	Display messages

## Header Specifications

### **Column 28**          **output option for Record-out sort**

Column 28 only applies to Record-out sorts (SORTR). It indicates whether or not you want to drop control fields from output records after they are sorted.

An X means drop them; leaving Column 28 blank means keeping the control fields.

### **Reasons for Dropping Control Fields**

If you are using an alternate collating sequence or opposite control fields you would probably drop the control fields. In each case the control information is so altered during the sorting process that it is of no use.

You can, however, output those fields in a meaningful form by describing them twice: once as control fields and once as data fields. Since data fields are not involved in the sorting process, they are not changed by the program.

### **Column 29-32**          **output record length for Record-out sorts**

Columns 29-32 apply to Record-out (SORTR) jobs only. The entry in these columns tells the program the length of records in the final sorted file.

When control fields are not being dropped, the length must include both control and data field lengths. Include only the data field lengths when the control fields are being dropped. Your total output record length, including control fields whether they are dropped or not, should not exceed the maximum work record length limit of 1000 bytes.

### **Calculating Output Record Length Fields**

When you are dropping the control fields, total the lengths of the data fields included in the job for each record section (see table on page 2-3). Select the largest total and enter the number in columns 29-32.

When you are not dropping the control fields, total the lengths of the data fields for each record section. Select the largest total and sum this with the entry in columns 13-17 of the Header before entering into columns 29-32.

### **Column 34-39**          **reserved for future use**

### **Column 40-80**          **job description comments**

Columns 40-80 are reserved for your comments. You can use any characters you want to in these columns. If your column 27 entry calls for the printing of specification lines, your comments here will be printed. Your comments have no effect on your program.

## ALTERNATE COLLATING SEQUENCE SPECIFICATIONS

SECTION

III

Normally, XSORT uses the standard ASCII collating sequence included in Appendix C.

If you want to change the standard collating sequence, you must code as S in column 26 of the Header Specification and code a set of ALTSEQ specifications. You can code whatever number of ALTSEQ lines you need. Each of your statements must begin with ALTSEQ and contain no more than 80 column entries per line.

Users accustomed to a 96 column line will have to convert their programs to the 80 column format. However, it is not necessary to fill in all 80 columns of one ALTSEQ line before continuing on to the next. As long as you enter the ALTSEQ in the first six columns, and submit your characters in pairs as described below, you can enter ALTSEQ characters without filling out all 80 column positions of a line. (The specification forms show only 72 columns but 80 columns per line are allowed.)

If you specify an alternate collating sequence, you cannot use packed data for record selection factors or sort control fields. Do not enter a P in column 8 of your Record Type specifications or in column 8 of your Field Description specs used to define control fields.

# Alternate Collating Sequence

## Rules for Coding ALTSEQ

1. Code ALTSEQ in the first six positions. This tells XSORT that this is an ALTSEQ specification.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3		
A	L	T	S	E	Q																													

2. Leave the two positions, columns 7 and 8 following ALTSEQ, blank.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3		
A	L	T	S	E	Q																													

3. Enter the hexadecimal equivalent of the character you are taking out of normal sequence in columns 9 and 10. (A standard collating chart showing the hex equivalents of all ASCII characters appears in Appendix C.)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3		
A	L	T	S	E	Q			M	C																									

4. Enter in 11 and 12 the hexadecimal equivalent of the value (position) the character specified in columns 9 and 10 is to have in the new collating sequence.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3		
A	L	T	S	E	Q			M	C	N	V																							

5. Enter as many such pairs as the number of characters you are taking out of normal sequence.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3			
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V

6. Leave no spaces between sets of hex numbers.
7. When you reach the end of a statement, continue on the next specification line, following the rules above (1-6). Entries can extend to column 80, although it is not necessary to fill an entire line before continuing to a new line.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3			
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V

8. Enter a double asterisk (\*\*) in columns 1 and 2 of the specification line. This indicates that you are finished with your ALTSEQ statements.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	3			
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V
A	L	T	S	E	Q			M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V	M	C	N	V
**																																			

\*MC=moved character, NV=new value

### Specs Order Using ALTSEQ

When you use an alternate collating sequence your input must follow an established order:

1. MPE commands
2. XSORT Specifications
  - a. Header
  - b. ALTSEQ entries (including the double asterisks \*\*)
  - c. S Option, if used
  - d. Record Type or Field Descriptions as they are required.

### Programming with ALTSEQ

Placing an alternate character in the sequence position which is normally assigned to another means that both occupy the same position and are considered equal. (See Sample on page 3-4.)

If you do not want to have the characters regarded as equal, the one which normally occupies that position must be moved.

### Effect of ALTSEQ Statements on Other Coding

ALTSEQ statements can change:

1. Factor 1 and factor 2 (See Record Type Specifications, Section 5.)
2. Normal and opposite control fields
3. Characters in control fields before they are replaced by forced fields

ALTSEQ statements do not change data fields in records or forced control field characters.

## Alternate Collating Sequence

### Sample ALTSEQ Statements

1. If you wished to make blanks and zeros equal in your collating sequence you would enter the following:

ALTSEQ 2030

Blank (20) is moved to the position occupied by zero (30). Blanks and zeros are, therefore, considered equal.

2. If you wished to change the sequence of several characters you should do the following:

ALTSEQ 2A4545464647

2A45 means \* (2A) will occupy the position normally occupied by E (45.).

4546 means E (45) will occupy the position normally assigned to F (46).

4647 means F (46) will occupy the position normally assigned to G (47).

The collating sequence, from character “)” through “H”, would be changed:

from: )\*+ . . . . . ABCDEFGH

to: ) + . . . . . ABCD\*E<sup>F</sup><sub>G</sub>H

In this example, F and G would occupy the same position.

The Special Option Specification presents you with additional runtime options. Special Option, referred to as S Option, can be thought of as an extension of your Header Specification. It offers global controls including:

- Address-out file first record number
- Inplace sort override
- Maximum number of records to be sorted

S Option is entered following your Header Specification and ALTSEQ specifications ( if used). It can be omitted if you have no need to exercise any of the options, in which case all options default to blank entries. However, it must be used when your input is non-disc or multiple file (to specify the maximum number of records to be sorted).

OVERVIEW OF COLUMN ENTRIES

COLUMN	ENTRIES	EXPLANATION
1-5	Any value	Line number
6	S	S Option will be exercised
7	0 or blank	Numbering for Address-out file starts at 0*
	1	Numbering for Address-out file starts at 1*
8	N or blank	XSORTOUT must be new or be the same file as XSORTIN
	Y	XSORTOUT may exist and be a separate file from XSORTIN
10-16	1-9999999 or blank	Maximum number of records to be sorted
17-39	blank	Reserved
40-80	Any Characters	Comments

*\*If you are sorting a KSAM file built with Firstrec=1, then this column is ignored and record numbering in the Address-out file automatically starts with 1.*

COLUMN ENTRIES

**Columns 1-5**      line number

Line number is an optional entry. XSORT does not check your columns 1-5 entries for sequence. Your entry here is intended as convenience, a record you may use for future reference.

**Column 6**      S entry

This specification type entry informs XSORT that you intend to use the S Option.

**Column 7**      Address-out file first record number

This entry states the first record number of an Address-out file. It is used primarily for sorts converted from other systems, such as IBM, that start record numbering at 1 instead of 0 (as HP does).

1. If you enter "0" or blank, the record numbering starts at 0.
2. If you enter "1" the record numbering starts at 1.



**Column 8**      **inplace sort override**

This entry tells XSORT whether or not you want to override the inplace sort rule.

Normally, XSORT protects you from accidentally overlaying an existing file with the sort output, XSORTOUT. To do this, it enforces the rule that XSORTOUT, if it already exists, must be the same file as XSORTIN, and this must be an inplace sort.

Your column entries determine whether or not you wish the rule to apply in the sort you are running.

<b>Column 8 entry</b>	<b>XSORTOUT file already exists</b>	<b>XSORTOUT file does not exist</b>
N or blank	XSORTOUT must be the same file as XSORTIN. (This is an inplace sort.)	New file will be created for XSORTOUT. It will be a job/session temporary file.
Y	XSORTOUT may be a different file than XSORTIN. It will be overlaid by sort output records.	

**Columns 10-16**      **maximum number of records to be sorted**

Here you enter the maximum number of records to be sorted. Your entry can range from 1 to 9999999.

If omitted, this value defaults to the number of records in the input file. An entry here is required if this is a multiple file sort or if input is from a non-disc device.

It is recommended that you specify this entry in cases where you expect only a small subset of records to be selected for processing from a large input file. This reduces the size of the SORT/3000 work file and results in more efficient use of disc space.



## RECORD TYPE SPECIFICATIONS

SECTION

V

Record Type Specifications tell XSORT which records in a file you wish to have sorted. If you want all the records in a file to be sorted, and they have all been prepared in the same format, you do not have to fill out this section of your XSORT specifications. Often, however, you use these entries to select records for the sort based on comparisons. Your Record Type Specifications will instruct XSORT to compare data in one field (factor 1) with a constant value or data in another field on the same record (factor 2) to decide whether to include (select) or omit (reject) the record for processing.

## OVERVIEW OF COLUMN ENTRIES

COLUMNS	ENTRIES	EXPLANATION
1-5	Any value	Line number
6*	I	Include line
	O	Omit line
7	A	AND line (these specifications continue the definition of the record which was described on the previous line)
	O	OR line (these specifications define a record different than the one described on the previous line)
	Blank *	First line of a set of I or O Record Type lines Comment lines
8*	C	Character equals 1 byte (eight bits) of data
	P	Packed decimal data
	U	Unpacked decimal data
9-12	1-1000	The input record position in which the factor 1 field begins (blank if the field is only one position long)
13-16*	1-1000	The input record position in which the factor 1 field ends
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*	C	Factor 2 is a constant
	F	Factor 2 is another field in the same input record
20-23	1-1000	The input record position in which the factor 2 field begins (blank if the field is only one position long)
24-27	1-1000	The input record position in which the factor 2 field ends
20-80	Any characters	The factor 2 constant
40-80	Any characters	Comments

\*Columns that must be filled in.

## COLUMN ENTRIES

### Columns 1-5      line number

Line number is an optional entry. XSORT does not check your columns 1-5 entries for sequence. Your entry here is intended as a convenience, a record you may use for future reference.

### Column 6      spec type

Your column 6 entry identifies the type of Record Type line you intend to use in your sort. An I in the column stands for an include or include-all line. An O stands for an omit line.

Include lines identify the records you want the program to sort by describing particular record fields.

If your sort requires the coding of Record Type lines, you must use at least one include or an include-all line to describe the records you want sorted. Even when you use omit lines, you will have to follow them with an include line. Records not described by your include lines will not be sorted.

Omit lines identify records you do not want the program to sort. Omit lines are not required but they can be helpful when you have many types of records you want to use and just a few you want omitted. Omit lines must be followed by at least one include line or an include-all line. This ensures that the program will check all records not described by an omit line.

## Record Types

Data records from the input file(s) may be identified by "type" according to characteristics defined in XSORT Record Type specifications. All records in any Record Type have one or more unique characteristics in common – for example, a 3 in position 5 AND not an X in position 80. Multiple characteristics used to identify one Record Type are tied together using the AND operation ("A" in column 7 of Record Type specifications).

## Sections

A "section" is a subset of the input records which is defined by naming one or more Record Types to be included in and/or omitted from a section. All records included in a section will be sorted and formatted in the same manner, according to Field Description specifications. The term "section" also refers to the XSORT specifications used to define the section. These consist of Record Type lines which identify the Record Type(s) of the section, followed immediately by Field Descriptions, showing how the section is to be sorted and formatted on output. These sections of specs (one or more) must all follow the Header, ALTSEQ, and S Option specifications.

Multiple Record Types named for one section are tied together using the OR operation ("O" in column 7 of Record Type specifications) whenever several include sets or several omit sets are defined consecutively. (See discussion of sets below.)

The specification sections are used in order of their appearance to identify and categorize each input record. The first section to "accept" an input record for processing is the one used to determine how it will be sorted and formatted. If a section "rejects" the input record, XSORT moves on to the next section to attempt identification. If no section "accepts" the record, it will not be included in any sort processing.

## Record Type Specifications

*Include-all* is a section of specifications consisting of a single Record Type line containing "1" in column 6 and blanks in columns 7 through 39, followed by Field Description lines. This says that all remaining input records are to be included in this section unconditionally. If used, it must be the last line of the last section defined for the sort job. Include-all is generally used as a "catch-all" mechanism to identify any records not already accepted for processing by any of the preceding specification sections.

*Implied Include-all* is a section of specifications consisting only of Field Descriptions. When used, it must be the only specification section defined for the job. It says that all input records are to be included in the sort unconditionally.

### Sets

The unique identification of a Record Type is made using sets of "include" and "omit" Record Type Specifications. These sets may be used singly or intermixed to include and/or omit data records in the current section according to their characteristics. (Multiple characteristics within a set must be tied together using the AND operation.)

*An Include set* defines the characteristics of a Record Type that is to be included (accepted) in the current section. (Consecutive include sets must be tied together using the OR operation on the first line of the set.)

*An Omit set* defines the characteristics of a Record Type that is to be omitted (rejected) from the current section. (Consecutive omit sets must be tied together using the OR operation on the first line of the set.)

Within a section, include/omit sets are used in order of their appearance to evaluate the current input record. If the record matches the characteristics defined for a set (Record Type), it is immediately "accepted" or "rejected" for inclusion in the section, and so no more include/omit sets will be evaluated. If the record does not match the set's characteristics, XSORT moves on to the next include/omit set for evaluation.

Additional information concerning record types, sets and sections and an example of how they are used appear in Section 8. This information is primarily useful for advanced sorts. Typically, users are concerned with simple identification of different Record types for inclusion or omission.

Column 7 continuation or comments (concerning column 6)

Column 7 indicates the start or continuation of a Record Type include or omit set definition.

COL. 7 ENTRY	EXPLANATION
Blank	This line contains the first characteristic of a new Record Type that has a different spec type (column 6) than the previous line.
A	This line contains additional characteristics for the same Record Type as that on the previous line. The A stands for AND.
O	This line contains the first characteristic of a new Record Type that has the same spec type (column 6) as the previous line. The O stands for OR.
*	The asterisk indicates a comment line. Comment lines are included as an aid in remembering why you were proceeding as you were at various points in the program. The comments will be printed if you leave column 27 of the Header specification blank or enter a zero.



The current Record Type can be further described using additional AND lines, or a new Record Type can be started after this one using an OR line or a "BLANK" line (in column 7), provided all Record Types are in the same section.

## Record Type Specifications

### Relating Columns 6 and 7 Entries

Because your column 6 and 7 entries often combine to complete include-omit specifications, the information below is intended to show how to use combinations for the two entries.

INCLUDE SETS*			
TYPE	COL. 6	COL. 7	EXPLANATION
Include AND lines	I	blank	First characteristic of a new Record Type. (Spec type different than previous line.)
	I	A	Additional characteristics for the same Record Type as the previous line.
Include OR	I ,	....	One or more lines defining the characteristics of one Record Type.
	I	O	First characteristic of a different Record Type than the previous line.
Include AND and Or lines	I	....	One or more lines defining the characteristics of one Record Type.
	I	O	First characteristic of a different Record Type than the previous line.
	I	A	Additional characteristics for the same Record Type as the previous line.
Include-all line	I		Sort all of the records that have not been described by any preceding include or omit sets. Only one include-all set may be defined per sort job.
<p><i>*An include set can be preceded by the Header (or ALTSEQ or S Option, if used), Field Descriptions for the previous section, or an omit set.</i></p> <p><i>An include set can be followed by another include set, an omit set, or Field Descriptions for the current section.</i></p>			



OMIT SETS*			
TYPE	COL. 6	COL. 7	EXPLANATION
Omit AND lines	O	blank	First characteristic of a new Record Type. (Spec type different than previous line.)
	O	A	Additional characteristics for the same Record Type as the previous line.
Omit OR line	O	...	One or more lines defining the characteristics of one Record Type.
	O	O	First characteristic of a different Record Type than the previous line.
Omit AND and OR lines	O	...	One or more lines defining the characteristics of one Record Type.
	O	O	First characteristic of a different Record Type than the previous line.
	O	A	Additional characteristics for the same Record Type as the previous line.
<p><i>*An omit set can be preceded by the Header (or ALTSEQ or S Option, if used), or Field Descriptions for the previous section, or an include set.</i></p> <p><i>An omit set can never be followed by Field Description lines. It must be followed by an include set so that the program will know which of the remaining (non-omitted) records you wish to have sorted.</i></p>			

### Mixing Sets

You may find occasions when mixing include and omit sets would be helpful. The program readily accepts the mix, but caution should be exercised since XSORT will accept and respond to commands in the order they are coded. You may, for instance, wish to omit records with a 1 in position 3 while including those with a 1 in positions 3 and 5. You would have to specify your include set first. If you began with your omit set, you would eliminate all the records you wished to sort.

## Record Type Specifications

### Column 8            data type (C/P/U)

The Column 8 entry tells XSORT how to interpret data during the compare operations. To do this, the program must first be told the format of the data that is to be compared.

XSORT accepts data in three formats:

- Character (C)            A character is an 8-bit pattern (byte) which represents a single alphabetic, numeric, special, or control symbol based on the American Standard Code for Information Interchange (ASCII).
- Packed (P)              Packed numerical data is compacted by storing two digits in each byte. Each digit is stored as a 4-bit binary number with the sign represented by a 4-bit sign code placed in the rightmost, low order position in the field.
- Unpacked (U)            Unpacked numerical data is represented by a subset of the full ASCII character set, consisting only of the numeric characters (0-9) and the special sign characters for the rightmost digit of numeric fields – positive ( { ,A-1) and negative ( } ,J-R).

### Importance of the Column 8 Entry

Through your Record Type Specifications you tell XSORT which records you want sorted. You do so by instructing it to compare data in one field – factor 1 – against a constant or data in another field on the same record – factor 2. The result of the comparisons determines whether or not the record will be sorted. The sort program sees your data as nothing more than a series of electronic bits. It must be told how to interpret the data:

- whether the data is numeric or character
- if it is numeric, whether it is packed or unpacked

COL. 8 ENTRY	COMPARISON TYPE	MAXIMUM FIELD LENGTH*
C	Character	256 characters
P	Numeric data is packed	14 bytes (27 packed digits and sign)**
U	Numeric data is unpacked	28 characters (Unpacked digits)**

*\* Maximum field length applies to both factor 1 and factor 2 data.  
\*\* If you specify an alternate collating sequence, you cannot use packed data for record selection factors or sort control fields. Do not enter a P in column 8 of your include or omit Record Type Specifications or in column 8 of your Field Description specs used to define control fields.*

### NOTE

See Appendix C for charts which will be helpful in data formatting.

**Columns 9-16      factor 1 field length**

Factor 1 field identifies your record. For example, all your inventory records may contain a 2 in column 2. That would make column 2 a factor 1 field. By comparing factor 1 fields against constants or other fields in the same record, XSORT identifies records you want sorted. Columns 9-16 identify the locations of factor 1 fields in the records. You may at times have more than one factor 1 field. When there is more than one, you must do the following:

- Describe each field in a separate Record Type line.
- Put an A (for AND) in column 7 for all but the first field. This will identify those following as applying to the same Record Type.

**Length of the Factor 1 Field**

Each factor 1 field can contain from 1 to 256 characters. A factor 1 field cannot be longer than the records you are working with, however. The length of your input records determines how long your factor 1 field can be. Your column 8 entry also controls the maximum length of the factor 1 field.

COL. 8 ENTRY	MAXIMUM FACTOR 1 FIELD LENGTH
C	256 characters *
P	14 characters **
U	28 characters
<p><i>* If factor 2 is a constant, the length of the factor 1 field must not exceed 61 characters (see columns 20-80 for additional information).</i></p> <p><i>** When factor 1 is packed, the field may actually represent 27 decimal digits and a sign.</i></p>	

# Record Type Specifications

## Coding Rules

These specifications should be right-justified. The entry which tells where the factor 1 field begins – where to read from – should end in column 12. The entry which tells where factor 1 ends – where to read through – should end in column 16. It is not mandatory that the entries be right-justified, but it makes the specifications more readable and transportable.

If you have factor 1 fields that are only one character long, you can leave columns 9-12 (from) blank and enter the single number in columns 13-16 (to). The example below described the position of a factor 1 field which has a 2 entered in column 4.

RECORD TYPE																																																															
Line Number	Spec. Type (L or O) Continuation (A, O, or *)	C.P. or U	Factor 1		Rel.	Factor 2 (Field or Constant)		Comments (any characters)																																																							
			Location from	to		Location from	to	Constant (any ASCII characters)																																																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60				
						4												EQ																																													

## Columns 17-18 type of comparison

This entry tells XSORT what kind of comparison you want to make between factor 1 and factor 2. You will specify the type and location of factor 2 when you fill in columns 20-80 (if it is a constant) or 20-27 (if it is another field in the same record). When you specify Alternate Collating Sequence, XSORT modifies factors 1 and 2 before making comparisons.

COL. 17-18 ENTRY	MEANING
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

**Column 19**      **field or constant**

XSORT identifies records you want to sort by comparing the factor 1 field (specified in columns 9-16) with either a constant or another field in the same record. Column 19 tells the program whether it is a field or a constant that you wish to use in the comparison.

- C means a constant will be used
- F means a field will be used

When you put a C in column 19, you use columns 20-80 for the constant. When you put an F in column 19, you use columns 20-27 to identify the location of the factor 2 field in the record.

**Columns 20-27**      **factor 2 field**

The factor 2 field must be the same length as the factor 1 field and in the same record as the factor 1 field.

These columns identify the location of the factor 2 field. Columns 20-23 (from) tell XSORT where the field starts. Columns 24-27 (to) tell the program where the field ends.

**Coding Entries**

Both the "from" and "to" sections should be right-justified. That is, the "from" entry should end in column 23, and the "to" entry should end in column 27.

Sometimes fields are only a single character in length. When there is only one character, skip the 20-23 entry, leaving it blank. Enter the column number of the record position containing the character in columns 24-27. Again, your entry should be right justified.

**Columns 20-80**      **factor 2 constant**

If factor 2 is a constant, you use columns 20-80 to enter the constant you have selected. Normally, columns 40-80 are reserved for comments. They can, however, be used as necessary for constants longer than 20 characters or digits. The constant can be any arrangement of characters.

When you enter a constant containing character data, placing a C in column 8, your entry must be the same length as the factor 1 field and must begin in column 20. Although the XSORT specification form includes 72 columns, your entries can extend to 80 columns.

## Record Type Specifications

### Numeric Constant (Packed or Unpacked)

Numeric constants are always right-justified according to their length. They must be the same length as the factor 1 field. If the factor 1 field is packed, the length of the factor 2 field will be twice as long.

For example, if factor 1 was a six position field in the input record and factor 2 was the numeric constant 135, you would have to right justify your factor 2 entry to column 25. You would enter 135 in columns 23, 24, and 25. It would not be necessary to include leading zeros since XSORT reads blanks and zeros the same.

### RECORD TYPE

Line Number	Spec. Type (I or O) Continuation (A, O, or *) C, P, or U	Factor 1		Rel. EQ, NE, LT, GT, LE, GE	Factor 2 (Field or Constant)		Commer
		Location from	to		Location from	to	
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							

or

Line Number	Spec. Type (I or O) Continuation (A, O, or *) C, P, or U	Location from	to	Rel. EQ, NE, LT, GT, LE, GE	Location from	to	Commer
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							

### Unpacked Constants

Unpacked constants must be the same length as the factor 1 field. Suppose your factor 1 field takes up four positions. Your entry for factor 2 must do the same. If the factor 2 constant is the value 5, you would enter a 5 in column 23. You could leave 20, 21, and 22 blank or fill them with zeros.

Negative unpacked constants require special attention. The negative sign must be "overpunched" on the last digit of the constant.

CODING NEGATIVE CONSTANTS (UNPACKED)	
If last digit in constant is	You replace it with this character
0	} (right brace)
1	J
2	K
3	L
4	M
5	N
6	O
7	P
8	Q
9	R

**Packed Constants**

Packed numbers in this entry receive different treatment. If factor 1 contains a packed number, the length of the constant must be twice the length of the factor 1 field. This includes the sign, "+" or "-", which must be placed in the rightmost position of the constant after the last digit (negative signs cannot be "overpunched" on the last digit of the constant as they are for unpacked numbers).

For example, here is the entry required to make a sort in which the record must contain a packed negative 3 (-3) in positions 3 and 4, an unpacked negative 24 (-24) in positions 10-13, and an unpacked negative 11 in positions 17-22.

**RECORD TYPE**

Line Number 1 2 3 4 5	Spec. Type (I or O) Continuation (A, O, or *) C, P, or U	Factor 1		Rel. EQ, NE, LT, GT, LE, GE	Factor 2 (Field or Constant)		Commr
		Location from	to		Constant (any ASCII characters) Location from	to	
6	I O	3	4	EQC	3-	PACKED	-3
7	I A U	10	13	EQC	2M	UNPACKED	-24
8	I O U	17	22	EQC	1J	UNPACKED	-11

**Columns 40-80    comments**

Columns 40-80 are reserved for your comments. These comments will be printed along with your specifications if you instructed XSORT to do so. (Column 27 of your Header line must either be blank or contain a zero.) Some programmers find it convenient to write the names of records described in the Record Type Specifications here, so an area has been set aside in the comment section for that purpose.

Constants longer than 20 characters or digits will extend past column 40 into the comment area. They will be used correctly and not truncated at column 40. (See the discussion of factor 2 constants, columns 20-80.)

Your comments do not influence the operation of your program in any way.





# FIELD DESCRIPTIONS SPECIFICATIONS

SECTION

VI

Field Description Specifications tell XSORT how to sort your records and format them in the output file.

Field Description Specifications

OVERVIEW OF COLUMN ENTRIES

COLUMNS	ENTRIES	EXPLANATION
1-5	Any value	Line number
6	F	Field Description line identification
7*	N	Normal control field
	O	Opposite control field
	F	Forced control field
	D	Data field
	*	Comment line
8*	P	Packed decimal data
	U	Unpacked decimal data
	C	Character data
	V	Force a data character into the data field.
9-12	1-1000	Starting position of field in the record. May be left blank if the field is only one character long.
13-16	1-1000	End position of field in the record
17	Any character	Forced control fields only (the character you want to change)
18	Any character	Forced control or data fields only (the character you want to force into position)
19	Blank	Forced control field line is not a continuation of the preceding line
	Any character other than blank	Forced control line is a continuation of the preceding line
20-39	blank	Reserved
40-80	Any characters	Comments

\*Columns that must be filled in.

**COLUMN ENTRIES****Columns 1-5**      **line number**

Line number is an optional entry. XSORT does not check your columns 1-5 entries for sequence. Your entry here is intended as a convenience, a record you may use for future reference.

**Column 6**      **line type**

An entry of F identifies this as a Field Description line.

**Column 7**      **field type or comments**

The column 7 entry tells XSORT whether you are describing a control field, data field, or a comment line. If you are describing a control field, the entry tells the program how it is to be used. Entries in columns 7 and 8 interact and a separate column description section is included below to clarify this interchange of commands.

The box below concerns the column 7 commands only.

<b>COLUMN 7 ENTRY</b>	<b>MEANING TO PROGRAM</b>
N	This is a normal control field. The field will be sorted according to the sequence specified in column 18 of the Header Specifications.
O	This is an opposite control field. It tells the program to sort the data in the sequence opposite to that specified in column 18 of the Header Specifications.
F	This is a forced control field. The entries in columns 9-19 indicate how the control field is to be changed.
D	This is a data field (used for SORTR only).
*	This is a comment line.

**Control Fields (N, O, or F) Entries**

When your file has more than one Record Section:

- The number of control fields does not have to be the same for all Record Sections.
- The total lengths of all the control fields do not have to be the same for all Record Sections (it is the maximum of these lengths that is entered in Header Specification columns 13-17).

**Normal and Opposite Control Fields (N and O) Entry**

These are fields used by the program to sort your input records. They are fields normally taken from the input records, although you can alter or add to them using forced control fields. (See later sections for more information concerning the use of forced control fields.)

## Field Description Specifications

### Opposite Control Fields

At times you may wish to sort records so that some control fields are in ascending order and some are in descending order. You use opposite control fields to accomplish this.

Your Header Specification establishes the default sorting order. You will sort in ascending order if you have specified descending order in your Header Specification and called for an opposite control field. You will sort in descending order if your Header Specification called for ascending order and you enter O for an opposite control field.

XSORT changes your control fields while building the work records if you specify opposite control fields. There is little reason to retain these fields and usually they are dropped — by entering an X in column 28 of the Header line. If you wish to retain the original control fields in your output records in a meaningful form, repeat the information as a data field.

### Sequencing Information with Control Fields

You determine the sequence of records in your Record-out sorts and the record addresses in your Address-out sorts by the order in which you describe control fields in your field description entries. How this occurs is illustrated below.

Input Record Position		1	2	3	4	5	6	7	8
Input Record Number	0	3	1				9	2	
	1	3	8			2	1	1	
	2	1	1			1	0	0	
	3	3	8			3	8	2	
	4	3	1			4	7	6	
	5	3	8			2	0	9	
	6	3	1			3	2	2	
		Item Number			Number Ordered				

Suppose your file is in ascending order. (You have entered an A in column 18 of your Header line.) You have also specified that the file has a normal control field in positions 2-3. It has an opposite control field in positions 6-8.

The file, unsorted, might look like this.

This is how XSORT would sort and output the file:

Output Record Position		1	2	3	4	5
Output Record Number	0	1	1	1	0	1
	1	3	1	4	7	6
	2	3	1	3	2	2
	3	3	1		9	2
	4	3	8	3	8	2
	5	3	8	2	1	1
	6	3	8	2	0	9
		Item Number			Number Ordered	

XSORT uses the control field in columns 2-3 to sort the item numbers in ascending order. It then uses the second control field to sort out the number of units ordered in descending order within the units group.



XSORT SPECIFICATIONS

HEADER

Line Number	Spec. Type (H)	Type of Sort	Largest sum of control field lengths of any record section (in bytes)	Sequence (A or D)	Reserved	Alternate Coll. Seq. (S)	Print Option	Output Option (X)	Length of output record	Reserved	Comments (any characters)
1-6											Entries can extend to 80 columns →
7	H	SORTA, SORTR, or SORTC		6A			OX		5		

← Insert ALTSE0 specifications here (if used)  
 ← Insert S OPTION specification here (if used).

RECORD TYPE

Line Number	Spec. Type (I or O)	Factor 1	Rel	Factor 2 (Field or Constant)	Comments (any characters)
7	*	Location from 10 to 15	F or C	Constant (any ASCII characters) Location from 20 to 27	Record Name
8	*				ALL RECORDS INCLUDED IN SORT

FIELD DESCRIPTION

Line Number	Spec. Type (F)	Location	Record character	Substitute character	Continuation	Forced	Reserved	Field Name	Comments (any characters)
6	F	2	0						THIS CONTROL FIELD DROPPED THIS CONTROL FIELD DROPPED CONTROL FIELD DESCRIBED AS DATA CONTROL FIELD DESCRIBED AS DATA
7	F	6	0						
8	F	6	0						
9	F	6	0						

## Field Description Specifications

### Forced Control Fields

You can force a character into your control field by using forced control. There are special considerations you should keep in mind when you plan to use forced control fields.

- Only one character can be used in your forced control field.
- You can use either a conditional or an unconditional force.
- A force-all line must be preceded by one or more conditional force lines.

### Conditional Force

A conditional force can be used only when a particular entry appears in the record. It allows you to change that entry to an alternate value for sorting purposes only. For example, you might have records to sort, each with a one-position control field. If that control field character were an A and you wished to change it to a \$ for sorting, you would use conditional force. XSORT would respond to your instructions in the following way:

- build a work record from the input record



A 1-position control field (A) in input record is moved to first control field position in work record.

- change each A to a \$



There are two types of conditional forces:

- Conditional force on Normal or Opposite control field
- Stand-alone conditional force

### Conditional force on Normal or Opposite Control Field

This allows you to specify a 1-character normal or opposite control field and have it converted to an alternate value on the work record, depending on its value on the input record. To use this option, you must first specify the normal or opposite control field and immediately follow it by the conditional force line or lines. Each conditional force line must show the same input field location in columns 13-16 and must include a continuation character in column 19 to indicate it is a continuation of the specification of the normal or opposite control field. This force option does not increase the length of your work record. It only overlays an already defined position.

### Stand-alone Conditional Force

If your conditional force lines are not a continuation of the specifications of a normal or opposite control field (see discussion above), XSORT will place a hex FF (if the sort is ascending) or a hex 00 (if the sort is descending) in the next available work record position. It will use that position as if it were a normal or opposite control field. Processing of the conditional force specifications then continues, based on that new position. Each conditional force line in a stand-alone group must show the same input field location in columns 13-16, even though that input field is never moved into the work record.

All lines except the first must include a continuation character in column 19. A stand-alone force group increases the length of your work record by 1.

### Force-all

Force-all is a different type of conditional force. It works only when the control field of the input record does not contain a particular entry or entries. For this reason it is sometimes referred to as a "catch-all" operation.

A force-all line always follows a series of conditional force lines. The combined conditional and force-all abilities allow you to give XSORT instructions such as the following:

- If the control field is an A, place a 5 in the work record.
- If the control field is a J, place a 4 in the work record.
- If the control field contains neither a A nor a J, place a \$ in the work record.

Here you are using two stand-alone conditional forces and following them with a force-all.

If the control field were in position 12 of the input records, here is how you would enter Field Description specifications for the example just described:

Line Number					Spec. Type (F)	Type (N,O,F,D,or *)	P,U,C, or V	Location from   to								Forced				
																Record character	Substitute character	Continuation		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
					F	F	C							1	2	A	5			
					F	F	C							1	2	J	4	X		
					F	F	C									\$	X			
					F															
					F															
					F															
					F															

## Field Description Specifications

### Unconditional Force

Your unconditional force does not have to be present in the input records. You might wish to place a 5 in the first position of every output record. Using an unconditional force control field, you could tell XSORT to place a 5 in the first available control field of the work record.

Control field portion	5		Data portion
-----------------------	---	--	--------------

All other control fields would then follow the 5 that has been unconditionally forced. Assume the input records are in this format.

Data	Control Field A	Data	Control Field B	Data
------	-----------------	------	-----------------	------

Your work record would look like this:

5	Control Field A	Control Field B	Data
---	-----------------	-----------------	------

Forced control fields affect work and output records only. They do not change input records. The column entries for columns 9-16, 17,18, and 19 contain further information and examples concerning specifications for using forced control fields.

### Data Fields (D) Entry

Data fields are fields you want the program to include in the output records but not use in sorting those records. The entry applies only to Record-out sorts. Within the Field Description lines of each Record Type section, control field lines must be placed before data field lines.

When your file contains more than one Record Section:

- all Sections are not required to have the same number of data fields.
- output data lengths do not have to be the same for all Sections. XSORT will place blanks to the right of shorter records to equalize lengths up to the output record length defined in Header Specifications columns 29-32.

### Comment Lines (\*) Entry

Comment lines have nothing to do with the way your program operates. They provide you with an opportunity to document your work. You can code comments anywhere within the XSORT specifications, but they will be printed only if you left blank or entered a 0 in column 27 of your Header Specification blank or entered a 0.



## Column 8            C, P, U, or V

Column 8 entries tell XSORT what type of data is to be used in the work records.

COLUMN 8 ENTRY	DATA TYPE USED	MAXIMUM FIELD LENGTH
C	Character equals one byte (8 bits) of data	256 characters
P	Packed decimal data	14 bytes (27 packed digits and sign)
U	Unpacked decimal data	28 characters (unpacked digits)
V	Force a data character into the record	1 character

You can force characters into your data field by placing a V in column 8 and specifying the character you want forced into that position in column 18. The character will then be placed in the next available data field position of the work record.

## NOTE

Column 8 entries in the Field Description Specifications may seem similar to the column 8 entries you entered as Record Type descriptions. They should not be confused. Column 8 of the Record Type specifications helps select the records you wish to sort. Column 8 of the Field Description tells how you want data used in sorting and outputting your work records.

## Packed Control Fields (Normal or Opposite)

XSORT also changes the control fields while building its work record if you specify packed control fields. The X in column 28 of your Header Specification will drop these changed control fields on output. If you wish to retain them they should be entered again as data fields.

COLUMNS 7 AND 8 – COMBINED ENTRIES		
COL. 7	COL. 8	MAXIMUM FIELD LENGTH
N or O	P	14
	U	28
	C	256
F	C <sup>1</sup>	!
D	P	14
	U	28
	C	256
	V	1
*		Comment line
<i>1. For an unconditional force and a force-all line, column 8 must contain a C.</i>		

## Field Description Specifications

### Columns 9-16 field locations

Columns 9-16 locate the input record positions that contain the field. The starting position (from) is identified in columns 9-12. The end position (to) is identified in columns 13-16.

The order you impose in your Field Description Specifications will determine the order in which they will be located in the output records.

For example, suppose your input records look like this.

### Input Record

Field name	ITEM	PRICE	ITEMS IN STOCK	REORDER POINT
record position	1 9	10 14	15 23	24 30

This is the way you want your output record to look

### Output Record

Field name	ITEM	REORDER POINT	ITEMS IN STOCK
2205			
record position	1 9	10 17	18 26

If you wanted to sort your records by item number, this is the way you would enter your Field Description Specifications.

### FIELD DESCRIPTION

Line Number	Spec. Type (F) Type (N,O,F,D, or *) P,U,C, or V	Location		Forced		Reserved	Field Name	Comments (any characters)
		from	to	Record character	Substitute character			
1	N	1	9				ITEM NUMBER	
2	D	10	14				REORDER POINT	
3	C	15	23				ITEM IN STOCK	
4								
5								

When you have a one-character control field, the procedure is the same with one exception. When the fields are only one character long, you leave columns 9-12 (from) blank and enter the number of the record position containing the character in columns 13-16.

As an example, suppose you wanted the 1-character field in position 12 of your input record to be in the first position of the sorted output record. You would describe the field in the first line of your Field Description Specifications, entering it as below

**FIELD DESCRIPTION**

Line Number	Spec. Type (F, P, U, C, or V)	Location	Reserved	Field Name	Comments (any characters)
		from to			
1	F	12			

Entries can extend to 80 columns →

Right-justifying entries is recommended. The starting (from) position of the input records ends in column 12. The end (to) position ends in column 16. In the example above, because it is a 1-character field, the 9-12 positions are left blank and the entry of the character's position is right-justified in column 16.

**Field Length**

The maximum allowable length of the field depends on your column 8 entry.

COL. 8 ENTRY	MAXIMUM FIELD LENGTH
P	14 characters
U	28 characters
C	256 characters
V	1 character

**Column 17 conditionally forced characters**

Forced control fields were generally discussed under the description of column 7 entries. In column 17 you make an entry only when you wish to use a conditional force; for example, if you wanted to place a \$ in a control field presently containing an A. Conditional force does not change the input record but it will replace each A with a \$ in that position in the work and output records.

Your entry in column 17 identifies the character that you want replaced in the control field. XSORT checks the input records to see if the control field contains the character you want replaced. Your entry in column 18 will then give XSORT the character you wish to have forced into that control field's position in the work record.

## Field Description Specifications

Sometimes control fields contain any one of several characters and you may wish to force a change for each of them. If this is the case, use a continuation character in column 19 (discussed below) on the specifications for all but the first of the forced characters.

### **Column 18**            **forced character**

The discussion of column 7 entries covers the subject of forced characters in detail. You use your column 18 entry only when you are using forced control fields, to sort your records, or forced data fields. In forcing a control field, the entry in column 18 either replaces the character you specified in column 17 or adds a character to the work record. If you are using the entry to force a data field, the character in column 18 will be added in the next available data field position of the work record.

#### NOTE

You can only use forced characters with 1-character fields.

Also, your input records are not changed when you use forced characters.

### **Column 19**            **continuation**

A non-blank character here says that this line is a continuation of the force character specifications group begun on the preceding line or lines.

When several force character specifications apply to the same 1-character control field, they must be entered together as a group. The first line of the group must have a blank in column 19 and all subsequent lines must have a non-blank character in column 19 to indicate continuation of the group of specifications.

## Summarizing Force Character Entries

### *Conditional Force Character*

1. Fill in columns 1-6 as you would for any control field.
2. Enter an F in column 7.
3. Enter C in column 8.
4. Leave columns 9-12 blank.
5. Specify the position of the control field in the input record in columns 13-16.
6. Enter the character you want replaced in column 17.
7. Enter your replacement character in column 18. Any character may be used.
8. Place any character in column 19 if this is the 2nd or more conditional force on the same control field. This will tell the program that this is a continuation of the preceding line.

### *Force-all Character*

1. Fill in columns 1-6 as you would for any control field.
2. Enter an F in column 7.
3. Enter C in column 8.
4. Leave columns 9-17 blank.
5. Put the replacement character for the control field in column 18.
6. Place any character in column 19. This will tell the program that this is a continuation of the preceding line.

### *Unconditional Force Character*

1. Fill in columns 1-6 as you would for any control field.
2. Enter an F in column 7.
3. Enter a C in column 8.
4. Leave columns 9-17 blank.
5. Put the character you are forcing in column 18.
6. Leave column 19 blank.

## Examples Using Forced Control Characters

Forced control characters can be used to add to or replace a character in a 1-position control field. Examples below show how to code four types of forced control characters:

*Unconditional force*

*Conditional force on normal or opposite control fields*

*Force-all*

*Stand-alone conditional force*

## Field Description Specifications

### Example 1: Unconditionally forced character

In this example a control field (\$) is unconditionally forced into the work and output records.

POSITION 1  
(Control Field)

INPUT RECORDS  
30 - 36 (Data Field)

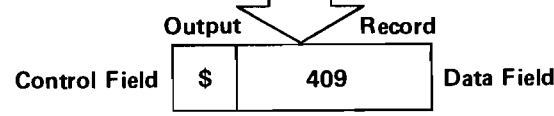
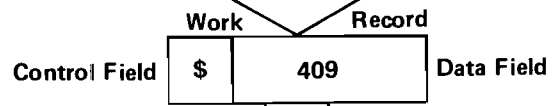


Field Description Specifications

Line Number	Spec. Type (F) Type (N,O,F,D,or *)	P,U,C, or V	Location		Forced														
			from	to	Record character	Substitute character	Continuation												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	F	F															\$		
	F	D						30				36							
	F																		
	F																		
	F																		
	F																		
	F																		
	F																		

The unconditional force moves a \$ into the first available position of the work record position.

Moves the data field (position 30-36 of the input record) to next seven positions of the work record.



## Field Description Specifications

### Example 2: Stand-alone conditional force

This example shows how different control field characters can be forced into position in the work record, depending on the character present in an input record position.

Here the column 19 entry is used to specify conditional forcing of more than one character. Because continuation lines are used in this example, only one position in the work and output records is defined by the first three lines. If column 19 were left blank and no continuation lines were used, each line would then define a new position in the work and output records.

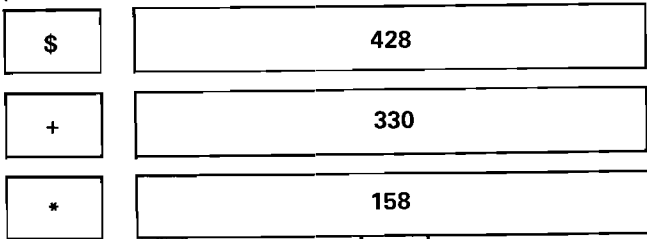
When you sort records in ascending order, having placed an A in column 18 of the Header, XSORT places hex FF into the work record before you force any characters. When you sort records in descending order, having placed a D in column 18 of the Header, the program places hex 00 into the work record before any characters are forced.



INPUT RECORDS

POSITION 7  
(Control Field)

18-20 (Data Field)



Field Description Specification

Line Number	Spec. Type (F) Type (N,O,F,D, or *) P,U,C, or V	Location		Forced		
		from	to	Record character	Substitute character	Continuation
1	F			7	\$	2
2	F			7	+	4
3	F			7	*	3
4	F					
5	F					
6	F					
7	F					
8	F					
9	F					
10	F					
11	F					
12	F					
13	F					
14	F					
15	F					
16	F					
17	F					
18	F					
19	F					
20	F					
21	F					
22	F					
23	F					
24	F					
25	F					
26	F					
27	F					
28	F					
29	F					
30	F					

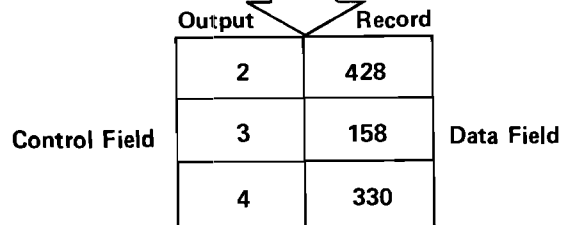
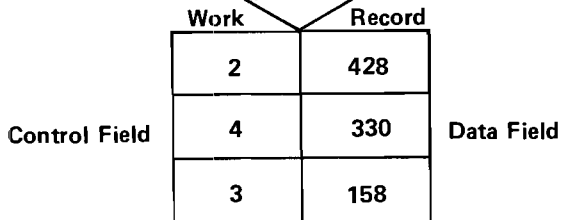
If position 7 of the input record contains a \$, move a 2 into the work record.

If position 7 of the input record contains a +, move a 4 into the work record.

If position 7 of the input record contains a \*, move a 3 into the work record.

If XSORT doesn't find a \$, +, or \* in position 7 of the input record, it will not change the hex 00 or FF which have already been placed in the work record.

Move the data field, positions 18-20, into the work record.



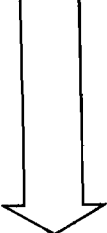
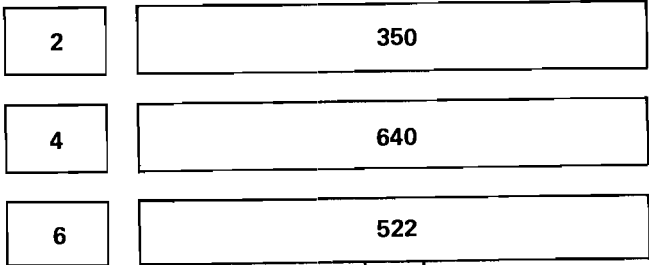
## Field Description Specifications

### Example 3: Conditional force on normal or opposite control field

In this example, as in the one previous, the control field in the input record changes the work and output records. This example is significantly different, however. Here the control field is moved to the work record and then changes are made. This occurs because the first control fields are normal control fields.

**POSITION 1  
(Control Field)**

**25 - 27 (Data Field)**



Line Number						Location											Forced		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
						Spec. Type (F)	Type (N, O, F, D, or *)										Record character	Substitute character	Continuation
						P, U, C, or V													
						from										to			
						H	C									1			
						F	C									1	4	6	X
						F	C									1	6	4	X
						D	C	2	5			2	7						

Move the control field -- in position 1 of the input record-- to the work record and use it to sort the records into ascending order, as specified by the header specifications.

If the control field contains a 4, replace it with a 6 in the work record.

If the control field contains a 6, replace it with a 4 in the work record.

Move the data fields -- positions 25 - 27 of the input record -- to the work record.

2	350
6	640
4	522



2	350
4	522
6	640

## Field Description Specifications

### Example 4: Force-all

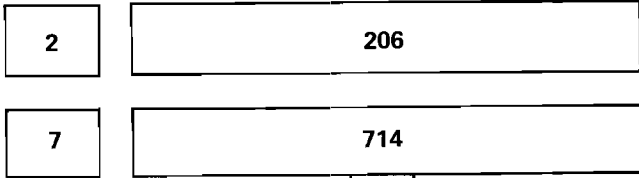
In the example, if control fields contain characters other than 2, 4, and 6, use a force-all line so that you won't have a hex FF or 00 present in the work and output records.

INPUT RECORD

POSITION 1

Control Field

24 - 26 (Data Field)



Field Description Specifications

Line Number	Spec. Type (F) Type (N,O,F,D,or *) P,U,C, or V	Location		Record character	Substitute character	Continuation	Forced
		from	to				
1	F			1	2	1	
2	F			1	4	3	X
3	F			1	6	5	X
4	F					\$	X
5	F						
6	F						
7	F						
8	F						
9	F						
10	F						
11	F						
12	F						
13	F						
14	F						
15	F						
16	F						
17	F						
18	F						
19	F						

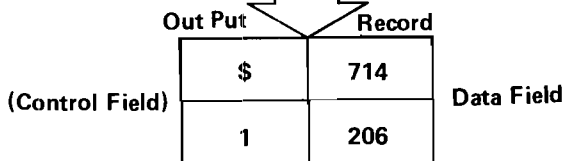
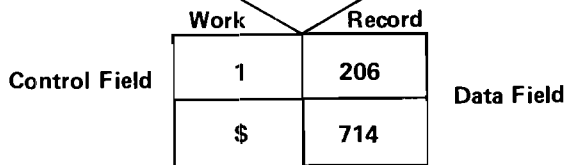
If position 1 of the input record contains a 2, move 1 into the work record.

If position 1 of the input record contains a 4, move a 3 into the work record.

If position 1 of the input record contains a 6, move a 5 into the work record.

If position 1 of the input record does not contain a 2, 4, or 6, move a \$ into the work record. (This is a force-all line.)

Move the data field - - positions 24-26 of the input record - - to the work record.



## Field Description Specifications

Columns 20-39 reserved

Columns 40-80 comments

These columns are reserved for your comments. Columns 40-45 are set aside by dotted lines as a convenient place to write down the names of fields described in the Field Description lines.

If you specified in your Header line that you wanted to have your specifications printed, your comments will be printed also. These comments have no effect on your program.

This section contains examples of the MPE commands necessary to run XSORT in interactive and job stream modes. These examples also encompass the various input and output file options provided by the utility.



## Using XSORT Commands

### RUNNING XSORT

Extra Function Sort for RPG/3000 can be run in job stream or interactively. Streaming is the recommended method because the program does not presently prompt for input.

On the following pages are examples of the MPE commands needed to run the program in several modes. The grid below indicates the alternatives open to you. The vertical column on the left presents the ways in which the specifications can be entered – either in job streams with specifications imbedded (or using the formal file designator, XSORTTEXT); or in sessions in which the data is entered interactively (or again, using XSORTTEXT). In the examples, the file equation XSORTTEXT=\$STDIN has been left out where it would otherwise be used as that is the default setting for XSORTTEXT.

The horizontal columns at the top of the grid show the choices open to you concerning input and output files as well as the Count-only sort (SORTC) designated Example #9.

Each box in the grid contains the number of the example (or combination of examples) to refer to for the sort to be run. To use the grid, you must determine what kind of sort you will require – whether it is an inplace sort or one using multiple input files, for example. Then your method of running the program – in job stream using XSORTTEXT, for example – must be selected. You can find the means by which this work can be successfully accomplished in the pages that follow and, if your needs are not explicitly covered, it is possible to combine the information included in the examples to enter your requirements into the system.

Single file input to different output (doesn't already exist)	Single file input – inplace	Single file input to different output (already exists)	Multiple input	SORTC (count-only)*
Job Stream: #1 Specs Imbedded	#6	#7	#8	#9
Job Stream: #2 XSORTTEXT	(Combine) #2 & #6	(Combine) #2 & #7	(Combine) #2 & #8	(Combine) #2 & #9
Session: Specs #3 Interactive	(Combine) #3 & #6	(Combine) #3 & #7	(Combine) #3 & #8	(Combine) #3 & #9
Session: #4 XSORTTEXT (Case 1)	(Combine) #4 & #6	(Combine) #4 & #7	(Combine) #4 & #8	(Combine) #4 & #9
Session #5 XSORTTEXT (Case 2)	(Combine) #5 & #6	(Combine) #5 & #7	(Combine) #5 & #8	(Combine) #5 & #9
Case 1 – XSORTIN and XSORTOUT are disc files. Case 2 – XSORTIN=\$STDIN and XSORTOUT=\$STDLIST. *Example #9 uses single file input. Count-only (SORTC) can also be used with multiple file input. To determine the appropriate commands for a multiple file SORTC, see Example #8 regarding how it specifies multiple input files (XSORTIN1. . . XSORTIN9).				



**About the sample sort:**

A very simple sort was devised for these examples. The input file, named TSTIN, consists of five records.

```

AB123GSD212345      JONES           X
AB234CIC012345      ALBERS
AC102GSD100001      WOODSON
AC321ABCD10001      FRANKEN
BB222CIC012345      WOODS
EOF FOUND IN FROMFILE AFTER RECORD 4

```

The specifications for most of the examples, whether entered by the user or supplied by the file, TSTSPEC, are these:

```

00000HSORTR          4A           X  33
00005I C             33NECX
00010FNC             6   9
00020FDC             1  32
00030FDV             *

```

In examples 5 and 8, it is necessary to include an S Option specification for reasons which are explained later. A file, TSTSPEX, which includes the S Option specs, was created for these examples. The entry in columns 10-16, which limits the "maximum number of records to be sorted", is set for Example 8.

```

00000HSORTR          4A           X  33
00000S Y             35
00005I C             33NECX
00010FNC             6   9
00020FDC             1  32
00030FDV             *

```

In Example 7, a file named TSTFILE, which exists previous to the sort, is overlaid by the sorted file. TSTFILE does not appear in the example as printed.



A brief explanation of the specifications may be helpful.

### Header Specifications:

COLUMN	ENTRY	EXPLANATION
1-5	blank	No line numbers are required.
6	H	The H designates this as the Header specification for the sort.
7-12	SORTR	SORTR indicates that this is a Record-out sort.
13-17	4	Largest sum of control field lengths for any record section, in bytes. In this sort the total is 4.
18	A	Records are to be sorted in ascending order by control field.
28	X	Control fields are to be dropped on output in this sort.
29-32	33	33 is the length of the output records in this sort. It includes only the length of the data fields since the control fields are being dropped.

### S Option Specifications

1-5	blank	No line numbers are required.
6	S	Indicates that this is the S Option specification for the sort.
8	Y	XSORTOUT may already exist and be a different file than XSORTIN.
10-16	35	Maximum number of records to be sorted.

### Record Type Specifications

1-5	5	Line number (optional).
6	I	This is an include line. All records described will be included in the sort.
8	C	Use full characters.
13-16	33	The factor 1 field, which is only one character long, ends at this position.
17-18	NE	Factor 1 must not equal factor 2.
19	C	Factor 1 is a constant.
20	X	This is the factor 2 constant.

### Field Description Specifications

1-5	1,2, and 3	Line numbers (optional).
6	F	These are field description lines.
7	N	This is a normal control field.
	D	This is a data field.
8	C	Use full characters in the field.
	V	Force a data character into the data field.
9-12	6, 1	These are the starting positions of the data fields in the records.
13-16	1, 32	These are the ending positions of the fields in the records.
18	*	This is the character you are forcing by using your column 8 entry.

## Using XSORT Commands

### About the Job Stream Examples

The easiest way to prepare job control commands is to enter them in a disc file using EDIT/3000. When you want to run a job stored in a disc file, you can use the MPE :STREAM command. All of the stream jobs described in the examples use this method.

When preparing a job to be streamed, you must substitute another character for the colon normally used with MPE commands. Any ASCII character may be used except the letters A through Z or the numbers 0 through 9. The character most commonly used is the exclamation point.

Although each of the job stream examples has unique characteristics, all have some in common. A general discussion of the commands for Example 1 will cover most of the common ground for the various sort configurations covered here. Where unique qualities are present, additional explanations will be appended to the example.

#### Example 1: Job stream with specifications imbedded

Single input to different output.

```
1  !JOB TSTXSRT1,MGR.SUBSYS,XSORT;OUTCLASS=,1
2  { !COMMENT *****
    !COMMENT *****
    !COMMENT THIS IS AN EXAMPLE OF HOW TO USE "XSORT" WITH
    !COMMENT SINGLE INPUT-TO-DIFFERENT OUTPUT
3  !FCOPY FROM=TSTIN;TO=
4  !PURGE TSTOUT
5  !FILE XSORTIN=TSTIN
6  !FILE XSORTOUT=TSTOUT
7  !RUN XSORT.PUB.SYS
8  { 0000HSORTR      4A          X  33
    00005I C        33NECX
    00010FNC      6   9
    00020FDC      1  32
    00030FDV          *
9  !EOD
10 !SAVE TSTOUT
    !FCOPY FROM=TSTOUT;TO=
11 !EOJ
```

1. The first command in a job must be the JOB command, followed by the user name, account name, and optionally the group name. In production work it is useful to specify a job name before the user name to identify the specific job that is being run. In this example, TSTXSRT1 is the job name.
2. Comment lines are used to describe the function or nature of the job.
3. The first program to be run in the job is FCOPY. It is included in order to show the contents of the input file prior to sorting. FCOPY is instructed to copy from the file TSTIN to the standard job/session device, \$STDLIST.
4. A file named TSTOUT is used for each of the jobs. It is purged initially to make sure that the name does not exist in the group.
5. The file referenced as XSORTIN is equated to the actual file named TSTIN.
6. The file referenced as XSORTOUT is equated to the actual file named TSTOUT.

7. The program XSORT.PUB.SYS is run.
8. These are the specifications for the sort. Note that they are not entered with the exclamation point since they are used as data by the XSORT program, instead of being used as MPE commands.
9. !EOD, entered with an exclamation point, indicates that you have entered all necessary data for the sort.
10. The output file TSTOUT is to be saved.
11. !EOJ signifies that you have completed the job entries.

A physically condensed copy of the example, after it was run, appears on page 7-8. It includes the output of the sort.

# Using XSORT Commands

```
:JOB TSTXSRT1,MGR.SUBSYS,XSORT
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J59
WED, MAR 11, 1981, 9:44 AM
HP3000 / MPE IV C.00.P8
  D U K E
COMMENT *****
:COMMENT *****
:COMMENT THIS IS AN EXAMPLE OF HOW TO USE "XSORT" WITH
:COMMENT SINGLE INPUT-TO-DIFFERENT OUTPUT
:FCOPY FROM=TSTIN;TO=
```

HEWLETT-PACKARD 32212A.3.13 FILE COPIER WED, MAR 11, 1981, 9:44 AM  
(C) HEWLETT-PACKARD CO. 1980

```
*200*
WARNING: FROMFILE RECSIZE IS 72 BYTES, TOFILE RECSIZE IS 133 BYTES.
```

```
AB123GSD212345    JONES          X
AB234CIC012345    ALBERS
AC102GSD100001    WOODSON
AC321ABCD10001    FRANKEN
BB222CIC012345    WOODS
EOF FOUND IN FROMFILE AFTER RECORD 4
5 RECORDS PROCESSED *** 0 ERRORS
```

```
END OF SUBSYSTEM
:PURGE TSTOUT
:FILE XSORTIN=TSTIN
:FILE XSORTOUT=TSTOUT
:RUN XSORT.PUB.SYS
```

HP32104A.05.00 EXTRA FUNCTION SORT FOR RPG/3000 (XSORT)  
(C) HEWLETT-PACKARD CO. 1980 WED, MAR 11, 1981, 9:44 AM

```
0000HSORTR      4A          X 33
00005I C        33NECX
00010FNC      6 9
00020FDC      1 32
00030FDV          *
```

```
RECORDS READ = 5
RECORDS SELECTED & SORTED = 4
  NUMBER OF COMPARES = 7
  SCRATCHFILE I/O = 2
  CPU SECONDS = .335
  ELAPSED SECONDS = 2.992
TOTAL CPU SECONDS = 1.006
```

```
END OF PROGRAM
:SAVE TSTOUT
:FCOPY FROM=TSTOUT;TO=
```

HEWLETT-PACKARD 32212A.3.13 FILE COPIER WED, MAR 11, 1981, 9:45 AM  
(C) HEWLETT-PACKARD CO. 1980

```
*200*
WARNING: FROMFILE RECSIZE IS 33 BYTES, TOFILE RECSIZE IS 133 BYTES.
```

```
AC321ABCD10001    FRANKEN          *
AB234CIC012345    ALBERS            *
BB222CIC012345    WOODS              *
AC102GSD100001    WOODSON            *
EOF FOUND IN FROMFILE AFTER RECORD 3
4 RECORDS PROCESSED *** 0 ERRORS
```

```
END OF SUBSYSTEM
:EOJ
CPU SEC. = 6. ELAPSED MIN. = 1. WED, MAR 11, 1981, 9:45 AM
```

## Example 2: Job stream with XSORTTEXT.

Single input to different output.

```

!JOB TSTXSRT2,MGR.SUBSYS,XSORT;OUTCLASS=,1
!COMMENT THIS IS AN EXAMPLE OF A SINGLE INPUT-TO-
!COMMENT DIFFERENT-OUTPUT SORT.THE JOB IS STREAMED,
!COMMENT THE SPECIFICATIONS ON XSORTTEXT.
!COMMENT*****
!COMMENT*****
!FCOPY FROM=TSTIN;TO=
!PURGE TSTOUT
!FILE XSORTIN=TSTIN
!FILE XSORTOUT=TSTOUT
!FILE XSORTTEXT=TSTSPEC
!RUN XSORT.PUB.SYS
!SAVE TSTOUT
!FCOPY FROM=TSTOUT;TO=
!EOJ

```

## NOTE

The only difference between Example 1 and Example 2 is the use of TSTSPEC in the equation using XSORTTEXT. TSTSPEC is an EDITOR file containing the same XSORT specs as those imbedded in the previous job stream.

## Using XSORT Commands

Example 3: Session with the specifications entered interactively.

Single input to different output.

```
:PURGE TSTOUT
:FILE XSORTIN=TSTIN
:FILE XSORTOUT=TSTOUT
:RUN XSORT.PUB.SYS
```

```
HP32104A.05.00      EXTRA FUNCTION SORT FOR RPG/3000      (XSORT)
(C) HEWLETT-PACKARD CO. 1980   WED, MAR 11, 1981,  9:16 AM
```

```
0000HSORTR      4A          X  33
0000HSORTR      4A          X  33
00005I C        33NECX
00005I C        33NECX
00010FNC      6    9
00010FNC      6    9
00020FDC      1   32
00020FDC      1   32
00030FDV          *
00030FDV          *
:EOD
```

```
RECORDS READ = 5
RECORDS SELECTED & SORTED = 4
  NUMBER OF COMPARES = 7
  SCRATCHFILE I/O = 2
  CPU SECONDS = .440
  ELAPSED SECONDS = 1.884
TOTAL CPU SECONDS = 1.779
```

```
END OF PROGRAM
:SAVE TSTOUT
```

1. After the command to run XSORT is given, the program prints an identification banner. When it is completed, the user enters the specifications for the session.
2. The specifications, which the user enters without prompting from XSORT, are echoed back to the terminal. (Note that the echoing can be eliminated by specifying a Print Option of 1, 2, or 3 in Header column 27.)
3. The user must supply the colon before the EOD.



Example 4: (Case 1) The first of two sessions using XSORTTEXT.

Single input to different output. Both XSORTIN and XSORTOUT are disc files.

```
:PURGE TSTOUT
:FILE XSORTIN=TSTIN
:FILE XSORTOUT=TSTOUT
:FILE XSORTTEXT=TSTSPEC
:RUN XSORT.PUB.SYS
```

```
HP32104A.05.00    EXTRA FUNCTION SORT FOR RPG/3000    (XSORT)
(C) HEWLETT-PACKARD CO. 1980    WED, MAR 11, 1981, 9:21 AM
```

```
00000HSORTR      4A          X 33)
00005I C         33NECX
00010FNC        6 9
00020FDC        1 32
00030FDV                *
```

} Specifications  
from TSTSPEC

```
RECORDS READ = 5
RECORDS SELECTED & SORTED = 4
NUMBER OF COMPARES = 7
SCRATCHFILE I/O = 2
CPU SECONDS = .339
ELAPSED SECONDS = 1.447
TOTAL CPU SECONDS = 1.116
```

```
END OF PROGRAM
:SAVE TSTOUT
```

## Using XSORT Commands

### Example 5:(Case 2) The second session using XSORTTEXT

Input from \$STDIN and output to \$STDLIST.

The file equation — XSORTIN=\$STDIN, in this example — means the program does not know in advance how many records are to be sorted since input is not from a disc file. As a result, it is necessary to enter an S Option line, specifying "Maximum number of records to be sorted" in columns 10-16.

```
:PURGE TSTOUT
:FILE XSORTIN=$STDIN
:FILE XSORTOUT=$STDLIST
:FILE XSORTTEXT=TSTSPEX
:RUN XSORT.PUB.SYS
```

```
HP32104A.05.00 EXTRA FUNCTION SORT FOR RPG/3000 (XSORT)
(C) HEWLETT-PACKARD CO. 1980 WED, MAR 11, 1981, 9:24 AM
```

```
00000HSORTR      4A      X  33)
00000S Y        35
00005I C        33NECX
00010FNC      6  9
00020FDC      1  32
00030FDV          *
AB123GSD212345  JONES
AB234CIC012345  ALBERS
AC102GSD100001  WOODSON
AC321ABCD10001  FRANKEN
BB222CIC012345  WOODS
:EOD
AC321ABCD10001  FRANKEN
AB234CIC012345  ALBERS
BB222CIC012345  WOODS
AC102GSD100001  WOODSON
```

Specifications from TSTSPEX

Interactive input from \$STDIN

Output to \$STDLIST

```
RECORDS READ = 5
RECORDS SELECTED & SORTED = 4
  NUMBER OF COMPARES = 7
  SCRATCHFILE I/O = 2
  CPU SECONDS = .000
  ELAPSED SECONDS = 101.440
TOTAL CPU SECONDS = 1.220
```

END OF PROGRAM

## Example 6: Job stream with specifications imbedded.

Single file, inplace sort.

```

!JOB TSTXSRT6,MGR.SUBSYS,XSORT;OUTCLASS=,1
!COMMENT THIS IS A SINGLE FILE, INPLACE SORT
!COMMENT THAT IS STREAMED. SPECS ARE IMBEDDED.
!COMMENT*****
!COMMENT*****
!FCOPY FROM=TSTIN;TO=
!FILE XSORTIN=TSTIN
!FILE XSORTOUT=TSTIN
!RUN XSORT.PUB.SYS
Specifications { 0000HSORTR      4A          X  33
imbedded in    00005I C      33NECX
in job stream  00010FNC    6    9
EDITOR file   00020FDC    1   32
              00030FDV      *
!FCOPY FROM=TSTIN;TO=
!EOJ

```

1. No purge of the XSORT file is required.
2. XSORTIN and XSORTOUT are both equated to TSTIN because this is an inplace sort.

## Using XSORT Commands

### Example 7: Job stream with specifications imbedded.

Single file to different output that already exists.

```
!JOB TSTXSRT7,MGR.SUBSYS,XSORT;OUTCLASS=,1
!COMMENT THIS IS AN EXAMPLE OF A SINGLE SORT FILE TO
!COMMENT DIFFERENT OUTPUT THAT ALREADY EXISTS.
!COMMENT IT IS STREAMED, SPECS IMBEDDED
!COMMENT *****
!COMMENT *****
!FCOPY FROM=TSTIN;TO=
!FCOPY FROM=TSTFILE;TO=
!FILE XSORTIN=TSTIN
!FILE XSORTOUT=TSTFILE
!RUN XSORT.PUB.SYS
0000HSORTR      4A          X  33
Specifications { 0000S Y
imbedded in   0000SI C      33NECX
job stream   0001OFNC      6   9
EDITOR file  0002OFDC      1  32
             0003OFDV          *
!FCOPY FROM=TSTFILE;TO=
!EOJ
```

1. No purge of the XSORTOUT file is required.
2. In this sort, the S Option must be employed to allow the program to overlay the existing file, TSTFILE.

#### NOTE

After this job is run, the old contents of TSTFILE will have been erased, and replaced by the same output data produced by the other examples.

## Example 8: Job stream with specs imbedded.

Multiple input file.

```

!JOB TSTXSRT8,MGR.SUBSYS,XSORT;OUTCLASS=,1
!COMMENT THIS IS A SORT USING MULTIPLE-
!COMMENT INPUT FILES. IT IS STREAMED,SPECS
!COMMENT IMBEDDED.*****
!COMMENT *****
!FCOPY FROM=TSTIN1;T0=
!FCOPY FROM=TSTIN2;T0=
!FCOPY FROM=TSTIN3;T0=
!PURGE TSTOUT
Multiple input { !FILE XSORTIN1=TSTIN1
file designators { !FILE XSORTIN2=TSTIN2
                   { !FILE XSORTIN3=TSTIN3
                   { !FILE XSORTOUT=TSTOUT
!RUN XSORT.PUB.SYS
Specifications { 00000HSORTR      4A          X  33
imbedded in   { 00000S Y          35
job stream    { 00005I C          33NECX
EDITOR file   { 00010FNC      6    9
               { 00020FDC      1   32
               { 00030FDV          *
!EOD
!SAVE TSTOUT
!FCOPY FROM=TSTOUT;T0=
!EOJ

```

1. S Option is used to provide the maximum number of records to be sorted.

## NOTE

The example uses XSORTIN1, XSORTIN2, etc. In a multiple input sort, XSORTIN cannot be used to designate an input file. Input files must begin with XSORTIN1 but may skip numbers — for example, XSORTIN3 could follow XSORTIN1.

## Using XSORT Commands

### Example 9: SORTC (Count-only sort)

Single input file -- no output

```
!JOB TSTXSRT9,MGR.SUBSYS,XSORT;OUTCLASS=,1
!COMMENT THIS IS AN EXAMPLE OF A COUNT-ONLY SORT FROM
!COMMENT WHICH THERE IS NO OUTPUT OTHER THAN A STATISTICAL
!COMMENT REPORT ON THE RECORDS SELECTED TO SORT.*****
!COMMENT*****
!FCOPY FROM=TSTIN;TO=
!FILE XSORTIN=TSTIN
!RUN XSORT.PUB.SYS
0000HSORTC
00005I C      33NECX }
!EOD          } Specifications
!EOJ          } imbedded in
                } job stream
                } EDITOR file
```

1. No purge of the XSORTOUT file is required.
2. No output file is designated since there is only a statistical report produced by a SORTC on the records selected for processing.

#### NOTE

Count-only (SORTC) can also be used with multiple file sorts. See Example 8 for uses of multiple input files (XSORTIN1. .XSORTING.) You can combine the commands in Example 8 with those above to produce a Count-only, multiple file sort.

Here is a physically condensed copy of Example 9 after it was run. Note the statistical report on the records sorted.

```

:JOB TSTXSRT9,MGR.SUBSYS,XSORT
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J58
WED, MAR 11, 1981, 9:40 AM
HP3000 / MPE IV C.00.P8
  D U K E
:COMMENT THIS IS AN EXAMPLE OF A COUNT-ONLY SORT FROM
:COMMENT WHICH THERE IS NO OUTPUT OTHER THAN A STATISTICAL
:COMMENT REPORT ON THE RECORDS SELECTED TO SORT.*****
:COMMENT*****
:FCOPY FROM=TSTIN;TO=

HEWLETT-PACKARD 32212A.3.13 FILE COPIER  WED, MAR 11, 1981, 9:40 AM
(C) HEWLETT-PACKARD CO. 1980

*200*
WARNING: FROMFILE RECSIZE IS 72 BYTES, TOFILE RECSIZE IS 133 BYTES.

AB123GSD212345      JONES          X
AB234CIC012345      ALBERS
AC102GSD100001      WOODSON
AC321ABCD10001      FRANKEN
BB222CIC012345      WOODS
EOF FOUND IN FROMFILE AFTER RECORD 4
5 RECORDS PROCESSED *** 0 ERRORS

END OF SUBSYSTEM
:FILE XSORTIN=TSTIN
:RUN XSORT.PUB.SYS

HP32104A.05.00      EXTRA FUNCTION SORT FOR RPG/3000      (XSORT)
(C) HEWLETT-PACKARD CO. 1980  WED, MAR 11, 1981, 9:40 AM

00000HSORTC
00005I C           33NECX

RECORDS READ = 5
RECORDS SELECTED = 4
TOTAL CPU SECONDS = .224 } Statistics

END OF PROGRAM
:EOJ
CPU SEC. = 4.  ELAPSED MIN. = 1.  WED, MAR 11, 1981, 9:41 AM

```

## Applications

The specifications for the sort are supplied by the file designated TSTSPECB.

HSORTR	13A	OX	80	EXAMPLE USING MIXED SECTIONS & SETS.
S Y	75			OVERLAY EXISTING OUTPUT; 75 RECS MAX
I*				
I C	6EQCX	} Include Set		SECTION-A, TYPE 1
IAC	23 24LEC50			(INCLUDE X'S BELOW 50)
IOC	6EQCY	} Include Set		SECTION-A, TYPE 2
IAC	23 24LEC50			(INCLUDE Y'S BELOW 50)
IOC	6EQCM	} Include Set		SECTION-A, TYPE 4
IAC	23 24LEC50			(INCLUDE M'S BELOW 50,
IAC	55 58GTF 51 54			AND ABOVE LIMIT)
F*				SECTION-A FIELD DESCRIPTIONS:
FNC	2 5	} Control Fields		SORT BY ITEM#, AND
FNC	1			CLASS
FDV				OUTPUT BLANK IN POSITION 1 } <i>Flags</i>
FDC	1 71	} Data Fields		OUTPUT DATA (FROM INPUT 1-71) } <i>Section A</i>
O*				
O C	6EQCZ	} Omit Set		SECTION-B, TYPE 3
OAC	12 13NECCA	} Include-all Set		(OMIT Z'S NOT IN CALIFORNIA)
I				SECTION-B, INCLUDE ALL OTHER RECORDS
F*				SECTION-B FIELD DESCRIPTIONS:
FNC	2 5	} Control Fields		SORT BY ITEM#,
FNC	1			CLASS, AND
FNC	73 80			LINE#
FDV				OUTPUT ASTERISK IN POSITION 1 } <i>Flags</i>
FDC	1 71	} Data Fields		OUTPUT DATA (FROM INPUT 1-71) } <i>Section B</i>
FDC	73 80			OUTPUT ORIGINAL LINE#'S } <i>Section B Only</i>

Because this is a multiple file sort, the S Option is used. The Y entered in column 8 indicates that you are willing to have the output file overlay an already existing file.

There are two sections, A and B, set off by the specifications. Section A is comprised of three Include Sets, plus control fields and data fields. A blank forced in position 1 of the output will "flag" Section A.

Section B is comprised of an Omit Set, an Include-all Set, and the control and data fields. An asterisk forced into position 1 of the output record "flags" Section B.

This is the input file, designated TSTINB, for the sort.



Item #		State	Size Code	Description	Limit Cost	Line #
Class	Group					
A0001X		CA	01	SECTION-A, TYPE-1	05001021	00001000
A0113Z		TX	21	SECTION-B, TYPE-3	09000899	00002000
A9059Z		CA	57	SECTION-B, OTHER	05002001	00003000
B0001X		TX	60	SECTION-B, OTHER	09000112	00004000
B0113M		FL	20	SECTION-A, TYPE-4	08000860	00005000
B9001M		CA	51	SECTION-B, OTHER	05000525	00006000
B9001M		CA	72	SECTION-B, OTHER	05000501	00007000
B9059Z		FL	01	SECTION-B, TYPE-3	08000801	00008000
C0001Y		CA	99	SECTION-B, OTHER	05000602	00009000
C0113Z		NY	20	SECTION-B, TYPE-3	02503000	00010000
C9001M		CA	99	SECTION-B, OTHER	05009999	00011000
C9001Y		OR	17	SECTION-A, TYPE-2	06000250	00012000
C9059Z		CA	10	SECTION-B, OTHER	05000980	00013000
C9059Z		CA	35	SECTION-B, OTHER	05000500	00014000
C9059Z		CA	87	SECTION-B, OTHER	05000970	00015000
D0001Y		FL	02	SECTION-A, TYPE-3	08000700	00016000
D0113M		TX	09	SECTION-B, OTHER	09000850	00017000
D9059Y		NY	50	SECTION-A, TYPE-2	02502140	00018000
W9001X		FL	13	SECTION-A, TYPE-1	08000750	00019000
1 - 6		12	23 26		51 58	73 80

Positions

The record portion designated "description" is used to highlight the purpose of the sort. Both sections will be sorted by item and class, and in Section B line numbers will also be used.

Here is the output file produced by the sort.

A0001X	CA	01	SECTION-A, TYPE-1	05001021	
*B0001X	TX	60	SECTION-B, OTHER	09000112	00004000
*C0001Y	CA	99	SECTION-B, OTHER	05000602	00009000
D0001Y	FL	02	SECTION-A, TYPE-3	08000700	
B0113M	FL	20	SECTION-A, TYPE-4	08000860	
*D0113M	TX	09	SECTION-B, OTHER	09000850	00017000
*B9001M	CA	51	SECTION-B, OTHER	05000525	00006000
*B9001M	CA	72	SECTION-B, OTHER	05000501	00007000
C9001Y	OR	17	SECTION-A, TYPE-2	06000250	
*C9001M	CA	99	SECTION-B, OTHER	05009999	00011000
W9001X	FL	13	SECTION-A, TYPE-1	08000750	
*A9059Z	CA	57	SECTION-B, OTHER	05002001	00003000
*C9059Z	CA	10	SECTION-B, OTHER	05000980	00013000
*C9059Z	CA	35	SECTION-B, OTHER	05000500	00014000
*C9059Z	CA	87	SECTION-B, OTHER	05000970	00015000
D9059Y	NY	50	SECTION-A, TYPE-2	02502140	

## NOTE

The file is in order first by item # (output positions 3-6), then by class number (output position 2), and then by line # (positions 73-80) for records in Section B. In addition, Section B records are "flagged" by an asterisk in output position 1 and by the input line # in positions 73-80.

## Applications

### Application #3

This application is in three parts. Each part is a sample job typical of the work XSORT was designed to address in the field. The jobs use the same files in a variety of ways for different sort requirements.

Each of the three sample jobs included is presented with an introduction, XSORT specifications entered on the form, and a discussion of the specifications.

A breakdown of the files to be sorted and their contents preceded the discussion.

FILES	RECORDS IN FILE	FIELDS USED IN SORTS
Weekly Sales	Detail sales records	Date, customer, product, quantity
Weekly bookings	Detail booking records	Date, customer, product, quantity
Sales and bookings	Summary sales and history bookings	Date, customer, product, quantity

### Record Format for Weekly Sales File

Field Names	Contents	Record Positions
RECCDE	Record Code "WS"	1 -2
INVNUM	Invoice Number	3 -8
INVDAY	Invoice data – day	9 -10
INVMON	Invoice data – month	11 -12
INVYR	Invoice data – year	13 -14
CUSTNO	Customer number	15 -24
CUSTTY	Customer type	25 -25
MRCHLN	Merchandise line	26 -27
PRODCO	Product code	28 -37
UNITS5	Sales units	38 -42
STACDE	State code	43 -44
SLSTER	Sales territory	45 -47
SLSNM1	Sales number	48 -49
TRANCD	Sales type	50 -50

## Record Format for Weekly Bookings File

Field Names	Contents	Record Positions
RECCDE	Record code "WB"	1 -2
CUSTNO	Customer number	3 -12
CUSTTY	Customer type	13 -13
ORDRNO	Order number	14 -19
ORTYCD	Order type	20 -20
MRCHLN	Merchandise line	21 -22
PRODCO	Product code	23 -32
STACDE	State code	33 -34
SLSTER	Sales territory	35 -36
SLSNM1	Salesman number	37 -39
CANIND	Cancel indicator	40 -40
DATEMM	Date – month	41 -42
DATEYY	Date – year	43 -44
UNITS5	Units booked	45 -49
TRANCD	Transaction code	50 -50

## Record Format for the Sales History File

Field Names	Contents	Record Positions
RECCDE	Record code "SB"	2 -2
CUSTNO	Customer number	3 -12
STACDE	State code	13 -14
SLSTER	Sales territory	15 -16
SLSNM1	Salesman number	17 -19
PRODCD	Product code	20 -29
CANIND	Cancellation indicator	30 -30
GARTCD	Garment type code	31 -31
DATEMM	Date – month	32 -33
DATEYY	Date – year	34 -35
UNITSB	Units booked	36 -42
UNITSHP	Units shipped	43 -49
UNTCAN	Units cancelled	50 -56
UNTRET	Units returned	57 -63
MRCHLN	Merchandise line	64 -65
DOLBOK	Booked value	66 -74
DOLSHP	Shipped value	75 -83
DOLCAN	Cancelled value	84 -92
DOLRET	Returned value	93 -101



### Sample Sort 1

This sort is used to select cancelled order records from the Weekly Bookings file. It produces a Weekly Cancellation Analysis. The "cancellation indicator" (in position 40), which is equal to 1, provides selection criteria.

Output records will contain only those fields required for the report. The output file will be sequenced by Customer Number (CUSTNO) and subsequently by Product Code (PRODCO).

Here are the specifications as they would appear at the terminal and on the specification form.

HSORTR		20A		34			SELECT CAN ORD.
I C		40EQC1					IND. = 1
I*							
FNC	3	12			CUSTNO		CUSTOMER #
FNC	23	32			PRODCD		PRODUCT CODE
FDC	13	13			CUSTTY		CUSTOMER TYPE
FDC	20	20			ORDTYP		ORDER TYPE
FDC	37	39			SLSNM1		SALESMAN NUMBER
FDC	41	42			DATEMM		DATE MOMNTH
FDC	43	44			DATEYY		DATE YEAR
FDC	45	49			UNITSS		ORDER UNITS



## Explanation of Specifications

### *Header Specification*

Column 6

The H entry identifies this as the Header line.

Columns 7-12

SORTR identifies this as a Record-out sort.

Columns 13-17

Total length of the sort control fields for this sort is 20. The total is derived from adding CUSTNO (10 characters) and PRODCO (10 characters).

Column 18

The A means the output sequence will be in ascending order by CUSTNO and PRODCO.

Columns 29-32

34 is the total length of output records. The control fields are not being dropped in this sort so their total (20 characters) is added to the 14 character sum derived from adding the six data fields in the sort.

### *Record Type Specifications*

Column 6

The I identifies this as an include line. Input records which match the criteria specified by this line will be used in the sort. Those which fail to meet the criteria will be omitted.

Column 7

The \* indicates a comment line.

Column 8

Use only complete characters — one byte (eight bits) — in making comparisons.

Columns 9-12

Because the input record field for factor 1 is only one character long, this entry is left blank.

Columns 13-16

Input record position where factor 1 field begins and ends is position 40.

Columns 17-18

EQ means factor 1 must equal factor 2.

## Applications

Column 19

The C means factor 2 is a constant.

Column 20

This is where the value of the constant is entered. In this instance, 1 is the factor 2 constant.

### *Field Description Specifications*

Column 6

The F identifies this as a field specification line.

Column 7

The N indicates that this is a normal control field. The D's in the column define the data fields.

Column 8

The C tells the sort to use complete characters — 1 byte (8 bits).

Columns 9-12

3, 23, and the other numbers indicate record positions where field start. They are the "from" positions.

Columns 13-16

These are the positions in the records where the fields end. They are the "to" positions.

Here is the output of the sort.

```
1111111111AAA555555555505018156789
2111111112DDD4444444446606028167890
2111111112XXX888888882202028123456
4111111114VVV666666664404118123457
4111111114ZZZ999999991101018112345
7111111117LLL777777773303128123456
EOF FOUND IN FROMFILE AFTER RECORD 5
6 RECORDS PROCESSED *** 0 ERRORS
```



## Sample Sort 2

This sort produces a record address file for the selected records of the Sales History File. Records are selected when they contain specific combinations of Sales Territory (SLSTER) and Merchandise Line (MRCHLN).

The output of this sort is a record address file containing relative record numbers for the selected records. The sequences will be Garment Type Code and Merchandise Line within the garment type. A hex dump of the output is followed by a simple RPG program which utilizes the output in a meaningful report.

Two additional tasks are performed here:

1. It is desired to combine the Garment Type Codes 2 and 3 into a single category for sorting purposes. Therefore, a "3" will be substituted for a "2" wherever it is encountered in the Garment Type code field. (This is a conditional force on the control field.)
2. An opposite sort is performed on units booked (UNITSB) to put the bookings in descending sequence within Merchandise Line.

Here are the specifications as they would appear at the terminal and on the specification form.

HSORTA		10A		
I C	15	16EQC01		SELECT TERR. 01
IAC	64	65EQC05		MERCH. LINE 05
IOC	15	16EQC01		OR
IAC	64	65EQC09		SELECT TERR. 01
I*				MERCH. LINE 09
FNC		31		FORCE TO EQ.CD.2
FFC		3123X	GARTCD	AND CODE 3
FNC	64	65	MRCHLN	MERCHANDISE LINE
FOC	36	42	UNITSB	UNITS BOOKED



## XSORT SPECIFICATIONS

### HEADER

Line Number	Spec. Type (H)	Type of Sort	Largest sum of control field lengths of any record section (in bytes)	Sequenced (A or D)	Reserved	Alternate Coll. Seq. (S)	Print Option	Output Option (X)	Length of output record	Reserved	Comments (any characters)
1-6											
7-12		SORTA, SORTR, or SORTC									
13-18											
19-39											
40-72											Entries can extend to 80 columns →
1	H	SORTA	10A								

← Insert ALTSEO specifications here (if used)  
← Insert S OPTION specification here (if used)

### RECORD TYPE

Line Number	Spec. Type (I or O)	Factor 1	Rel	Factor 2 (Field or Constant)	Comments (any characters)
1-5					
6-8		Location	EQ, NE, LT, GT, LE, GE	Constant (any ASCII characters)	
9-16		from to		Location	
17-19					Record Name
20-72					Entries can extend to 80 columns →
1	I	15	16	EQ	
2	I	64	65	EQ	SELECT TERRITORY 01 AND
3	I	15	16	EQ	MERCHANDISE LINE 05
4	I	64	65	EQ	OR
5	I	15	16	EQ	SELECT TERRITORY 01 AND
6	I	64	65	EQ	MERCHANDISE LINE 09

### FIELD DESCRIPTION

Line Number	Spec. Type (F)	Location	Forced	Record character	Substitute character	Continuation	Reserved	Field Name	Comments (any characters)
1-5									
6-8		from to							
9-16									
17-19									
20-72									Entries can extend to 80 columns →
1	F	31							
2	F	31	23	X					FORCE TO EQUATE CODE 2
3	F	64	65						GARTCD AND CODE 3
4	F	36	42						MRCHLN MERCHANDISE LINE
5	F								UNITSB UNITS BOOKED

## Explanation of Specifications

### *Header Specification*

Column 6

The H identifies this as the Header line.

Columns 7-12

SORTA identifies this as an Address-out sort.

Columns 13-17

10 is the largest sum of all control field lengths. It is derived generally from adding all control fields designated with an F, N, or O in column 7 for each record type. This case is an exception because a forced control field is being used. It will overlay the position in the work record occupied by the character it is forcing and, therefore, will not contribute to the total control field length.

Columns 18

The A indicates that the sorted records will be in ascending order by control fields.

### *Record Type Specifications*

Column 6

The I identifies this as an include line. The input records matching the criteria specified by this line will be used by the sort.

Column 7

Leaving this column blank indicates that this is the first line of an I (include) set of record type lines. The \* means a comment line.

A identifies an And line. It signifies that these specifications continue the definitions of the record type described in the last record type entry.

O entered here stands for Or. It means that these specifications define a different type of record than the one on the previous line.

Column 8

A "C" here means a character is used as comparisons.

Columns 9-12

The numbers 15 and 64, entered twice here, indicate the position where the factor 1 field begins.

Columns 13-16

The numbers 16 and 65 entered here indicate where the factor 1 field ends.

## Applications

### Columns 17-18

Factor 1 will be compared with factor 2, which is a constant in all cases here. EQ means you want the two factors to be equal.

### Column 19

The C identifies factor 2 as a constant.

### Columns 20-23

The entries 01, 05, 01 and 09 are two-character constants. The zeros are entered here because the control fields are all two characters long, and constants must be the same size as control fields. (You can leave the positions presently occupied by zeros blank. It is important, however, to keep the entries right justified.)

### *Field Description Specifications*

#### Column 6

The F identifies this as a field description line.

#### Column 7

The N identifies it as a normal control field.

O means opposite control field.

F indicates that one control field will be forced into position. In this case it will overlay the position in the work record occupied by the previous control field — if the conditions of the force operations are met.

#### Columns 9-12

The numbers entered in this column identify the starting position of a control field in a record.

#### Columns 13-16

These numbers identify the end position of a control field in a record. Note that 31, which is a field that is only one position in length, is only entered once, at the end position.

#### Column 17

The 2 is the character that must be in this control field in order for a 3 (defined in column 18) to be forced into the work record.

#### Column 18

The 3 is the character which will replace the control field in the work record if the control field equals 2 (defined in column 17).

Since this is a Address-out sort, a hex dump of the output is provided.

```
SRTUT RECORD 0 (%0, #0)
00000: 0000 0002
```

```
SRTUT RECORD 1 (%1, #1)
00000: 0000 0001
```

```
SRTUT RECORD 2 (%2, #2)
00000: 0000 000B
```

```
SRTUT RECORD 3 (%3, #3)
00000: 0000 0003
```

```
SRTUT RECORD 4 (%4, #4)
00000: 0000 0004
```

```
SRTUT RECORD 5 (%5, #5)
00000: 0000 0005
```

```
SRTUT RECORD 6 (%6, #6)
00000: 0000 0006
```

```
SRTUT RECORD 7 (%7, #7)
00000: 0000 0007
```

# Applications

A simple RPG program has been devised to make use of the output of the sort.

PAGE 0001 HEWLETT PACKARD 32104A.04.07 RPG/3000 (C) HEWLETT-PACKARD CO. 1978.  
 WED, MAR 11, 1981, 10:24 AM

0001 \$CONTROL USLINIT  
 0002 H

1

0003 FHISTORY IP F 101R I DISC  
 0004 FRAF IRE F 4 4 T EDISC  
 0005 FREPORT 0 F 132 OV LLP

0006 E RAF HISTORY

0007 LREPORT 66FL 600L

0008 IHISTORY NS 01  
 0009 I  
 0010 I 3 12 CUSTNO  
 0011 I 31 31 GARTCD  
 0012 I 32 350DATE  
 I 36 420UNITSB

0013 C TOTAL ADD UNITSB TOTAL 90

0014	OREPORT	H	201	1P		
0015	0	OR		OV		
0016	0				UPDATE Y	16
0017	0					51 "SALES HISTORY LISTING"
0018	0				PAGE Z	72
0019	0					70 "PAGE"
0020	0	H	1	1P		
0021	0	OR		OV		
0022	0					37 "GRMNT UNITS"
0023	0					58 "CUSTOMER DATE"
0024	0	H	1	1P		
0025	0	OR		OV		
0026	0					37 "-----"
0027	0					58 "-----"
0028	0	D	1	01		
0029	0				GARTCD	25
0030	0				UNITSBJ	38
0031	0				CUSTNO	50
0032	0				DATE Y	58
0033	0	T	1	LR		
0034	0				TOTAL J	38

Here is the output of the program. You will notice that the sequence under the heading "GRMNT" (GARMENT) seems to start in descending rather than ascending order. In fact, a 3 was substituted for each 2 when encountered in the Garment Type code field, and sorting was done according to these values.

3/11/81

## SALES HISTORY LISTING

PAGE 1

GRMNT	UNITS	CUSTOMER	DATE
-----	-----	-----	-----
3	3,456,789	3111111113	2/81
2	2,345,678	2111111112	2/81
2	2,222,221	2111111112	2/81
4	4,444,444	4111111114	1/81
5	5,555,551	5111111115	2/81
6	6,666,661	6111111116	2/81
7	7,777,771	7111111117	2/81
8	8,888,881	8111111118	2/81
	41,357,996		

## Applications

### Sample Sort 3

This sample sort utilizes the multiple input file feature of XSORT. It combines the Weekly Bookings file and the Weekly Sales file.

The sequence of the combined output file will be by customer number. Additionally, the Weekly Sales file is reformatted on output to the sorted file in order to match the format of the Weekly Bookings file records.

The file statements required for this sort are:

```
FILE XSORTIN1=WKBOOK
FILE XSORTIN2=WKSALE
FILE XSORTIN3=HISTUPD
```

Here are the specifications as they would appear at the terminal and on the specification form.

HSORTR	10A	X	90		
S	36				
I C	1	2EQCWB			SEL.WKLY.BK.REC.
FNC	3	12		CUSTNO	CUSTOMER NUMBER
FDC	1	50			
I C	1	2EQCWS			SELECT WEEKLY
I*					SALES RECORD
FNC	15	24		CUSTNO	CUSTOMER NUMBER
FDC	1	2		RECCDE	RECORD CODE
FDC	15	24		CUSTNO	CUSTOMER NUMBER
FDC	25	25		CUSTTY	CUSTOMER TYPE
FDC	3	8		INVNUM	INVOICE NUMBER
FDC	50	50		TRANCD	SALES TYPE
FDC	26	27		MRCHLN	MERCHANDISE LINE
FDC	28	37		PRODCD	PRODUCT CODE
FDC	43	44		STACDE	STATE CODE
FDC	45	47		SLSTER	SALES TERRITORY
FDC	48	49		SLSMN1	SALESMAN NUMBER
FDC	50	50		TRANCD	SALES TYPE
FDC	11	12		INVMON	INVOICEDATEMONTH
FDC	13	14		INVYR	INVOICEDATEYEAR
FDC	38	42		UNITSF	SALES UNITS





XSORT SPECIFICATIONS

HEADER

Line Number	Type of Sort		Largest sum of control field lengths of any record section (in bytes)	Sequence (A or O)	Reserved	Alternate Coll. Seq. (S)	Print Option	Output Option (X)	Length of output record	Reserved	Comments (any characters)
	SORTA, SORTR or SORTC										
1-5											Entries can extend to 80 columns →
6	SORTR			10A				X	90		

← Insert ALTSEO specifications here (if used)  
 ← Insert S OPTION specification here (if used)

S OPTION

Line Number	Spec. Type (S)	0 or 1	N or Y	Reserved	Maximum number of records to be sorted	Reserved	Comments (any characters)
1-4							Entries can extend to 80 columns →
5	S				36		

RECORD TYPE

Line Number	Spec. Type (I or O)	Continuation (A, O, or *)	Factor 1		Rel	Factor 2 (Field or Constant)		Comments (any characters)
			Location	Location		Constant (any ASCII characters)	Record Name	
1-5			from	to	EQ, NE, LT, GE, F or C	from	to	Entries can extend to 80 columns →
6	I	C	1	2	EQCWB			
7-11								SELECT WEEKLY BOOKING RECORDS

FIELD DESCRIPTION

Line Number	Spec. Type (F)	Type (N, O, F, D, or *)	P, U, C, or V	Location		Record character	Substitute character	Continuation	Forced	Reserved	Field Name	Comments (any characters)
				from	to							
1-5				from	to						Entries can extend to 80 columns →	
6	F	NC		3	12							
7	F	DC		1	50						CUSTNO CUSTOMER NUMBER	



XSORT SPECIFICATIONS

RECORD TYPE

Line Number	Spec. Type (I or O) Continuation (A, G, or *) C, P, or U	Factor 1		Rel	Factor 2 (Field or Constant)		Comments (any characters)
		Location from	to		Constant (any ASCII characters) Location from	to	
1	I	1	1	EQ	2	EQCWS	SELECT WEEKLY SALCT RECORD

FIELD DESCRIPTION

Line Number	Spec. Type (F) Type (N, O, F, D, or *) P, U, C, or V	Location		Record character Substitute character Continuation	Forced	Comments (any characters)
		from	to			
F	N	15	24			CUSTNO CUSTOMER NUMBER
F	D	1	2			RECCDE RECORD CODE
F	D	15	24			CUSTNO CUSTOMER NUMBER
F	D	25	25			CUSTTY CUSTOMER TYPE
F	D	3	8			INNVN INVOICE NUMBER
F	D	50	50			TRANCD SALES TYPE
F	D	26	27			MRCHLN MERCHANDISE LINE
F	D	28	37			PROCD PRODUCT CODE

FIELD DESCRIPTION

Line Number	Spec. Type (F) Type (N, O, F, D, or *) P, U, C, or V	Location		Record character Substitute character Continuation	Forced	Comments (any characters)
		from	to			
F	D	43	44			STACDE STATE CODE
F	D	45	47			SLSTER SALES TERRITORY
F	D	48	49			SLSNMI SALESMAN NUMBER
F	D	50	50			TRANCD SALES TYPE
F	D	11	12			INVMON INVOICE DATE MONTH
F	D	13	14			INVYR INVOICE DATE YEAR
F	D	38	42			UNITSS SALES UNITS

## Explanation of Specifications

### *Header Specification*

Column 6

The H identifies this as the Header line.



Columns 7-12

SORTR identifies this as a Record-out sort.

Columns 13-17

10 is the largest total of control field lengths for either of the two record sections in the sort. It is the sum derived from adding together all the control fields (fields with N, O, or F in column 7 of the Field Description specifications). In this case, control field length for the first record section is the same as that for the second section since it represents the same field, Customer Number. The field's location changes due to differences in the record formats of the two input files.

Column 18

The A indicates that the records in the sorted file will be in ascending order by control field.

Column 28

The X entered will cause the control fields to be dropped from the output records.

Columns 29-32

90 is the length of the output records in the Record-out sort job. In this sort, control fields are being dropped so the total reflects the largest total of data fields for any single record section. If the control fields were being included in the output file, their largest total (from Header columns 13-17) would be added to that of the data fields.

### *S Option*

Column 6

The S indicates that the S Option is being used. In this case, it is required because of the multiple input files.

Column 10/16

36 is the maximum number of records to be sorted.

### *Record Type Specifications (First Record Section)*

Column 6

The I entry identifies this as an include line.

## Applications

### Column 7

Leaving this blank identifies this as the first line of an include set.

### Column 8

C means the full character — 1 byte (8 bits) — will be used in comparisons done for this sort.

### Columns 9-12

1 is the position in the input record where the factor 1 field begins.

### Columns 13-16

2 is the position in the input record where the factor 1 field ends.

### Columns 17-18

EQ means that factor 1 must equal factor 2 in the comparisons.

### Column 19

The C identifies factor 2 as a constant rather than a field.

### Columns 20-39

WB is the factor 2 constant.

### *Field Description Specifications (First Record Section)*

### Column 6

F indicates that this is a field description specification.

### Column 7

N identifies a normal control field.

D identifies a data field.

### Column 8

C indicates that a full character — 1 byte (8 bits) — will be used.

### Columns 9-12

3 is the input record position where the control field begins. 1 is the input record position where the data field begins.

### Columns 13-16

12 is the input record position where the control field ends. 50 is the input record position where the data field ends.

***Record Type Specifications (Second Record Section)***

## Column 6

I indicates that this is an include line.

## Column 7

Leaving this entry blank means that this is the first line of an include set. The \* means comments.

## Column 8

C means the full character — 1 byte (8 bits) — will be used in the comparisons.

## Columns 9-12

A 1 entered here indicates the position in the record where the factor 1 field begins.

## Columns 13-16

2 entered here indicates the position in the record where the factor 1 field ends.

## Columns 17-18

EQ means that, in the comparisons, factor 1 must equal factor 2.

## Column 19

The C indicates that factor 2 is a constant, not a field.

## Columns 20-27

WS is the factor 2 constant. Note that this is the same length — 2 characters — as the factor 1 field.

***Field Description Specifications (Second Record Section)***

## Column 6

F indicates that this is a field description line.

## Column 7

N identifies this as a normal control field.

D identifies this as a data field.

## Column 8

C entered here means the full character — 1 byte (8 bits) — will be used in the comparisons.

## Applications

### Column 9-12

These entries identify the position in the input records where control and data fields begin. Note that Customer Number, which is used as a control field, is entered here again as a data field. Since control fields are being dropped from output records, they must be redefined as data fields in order to be included as output.

### Columns 13-16

Numbers entered in these columns identify the positions in the input records where the fields end.

This is the output of the sort.

```
WS01111111100900003009VVV0000000CA031100018101234
WB11111111115999995505AAA5555555CA0150510181567895
WB11111111119999989109ZZZ9999999CA31202N0281901233
WS11111111119900012905ZZZ9999999CA011019028112345
WS121111111121900002109ZZZ9999999CA031111018123456
WB211111111128999998205XXX8888888TX0120210281234568
WB211111111124999994605DDD4444444CA0160610281678904
WB211111111120999990009XXX0000000CA21101N0281901232
WS211111111128900011805XXX8888888CA011028028123456
WS231111111132900001209XXX8888888CA031122018134567
WS311111111137900010705LLL7777777TX011037018134567
WB411111111149999999105ZZZ9999999CA0110110181123459
WB411111111146999996405VVV6666666NV0140411181234570
WB411111111142999992809ZZZ2222222CA91808N0181890122
WB411111111148999988209ZZZ8888888CA41303N1281901234
WS411111111146900009605VVV6666666AZ011046018145678
WS511111111155900008505AAA5555555CA011055018156789
WS511111111153900006309JJJ3333333CA031073018178901
WS611111111164900007405DDD4444444CA011064018167890
WB711111111177999997305LLL7777777AZ0130311281234569
WB711111111173999993709LLL3333333CA81707N0281789013
WB711111111171999991909ZZZ1111111CA11909Y0181901231
WS811111111182900005209GGG2222222CA031082018189012
WS911111111191900004109ZZZ1111111CA031091018190123
```





### Compatibility with SORT3

XSORT is based on the unsupported utility program named SORT3. To avoid difficulties in converting from SORT3, XSORT has been designed to recognize the formal file designators used by SORT3.

There are no significant differences in the syntax of the sort specifications between the two products. The only issue to resolve in the conversion is the naming of the program file – XSORT vs SORT3. To do this, first use the command:

```
:LISTF SORT3.a.a,1
```

This will determine all the groups and accounts in which SORT3 has been stored. Then log on to each of those groups and do the following:

```
:PURGE SORT3  
:FCOPY FROM=XSORT.PUB.SYS;TO=SORT3;NEW
```

This will allow all old SORT3 job streams, UDC's, and operator procedures to work without requiring changes for XSORT.

## Compatibility

Correlation of XSORT and SORT3 formal names:

FILE	XSORT NAMES	SORT3 NAMES	CONVERSION ACTIONS
Program	XSORT.PUB.SYS	SORT3.grp.acct	:HELLO user.acct, grp :PURGE SORT3 :FCOPY .....**
Specifications	XSORTEXT	SPECCARD	No action necessary
Single input	XSORTIN	SORT3IN	No action necessary
Multiple input	XSORTIN1 . . . . XSORTIN9	SORT3IN1 . . . . SORT3IN9	(Note, however, that the first input file, ". . .IN" or ". . .IN1, " is what determines whether XSORT or SORT3 names will be expected for rest of the input/output files. No mixing of formal file names is allowed among the input and output files within any sort job.)
Output	XSORTOUT	SORT3OUT	
**The full command would be :FCOPY FROM=XSORT.PUB.SYS;TO=SORT3;NEW			

## Comparison of Extra Function Sort for RPG/3000 with IBM System/3 \$DSORT and IBM System 32 & 34 #GSORT

### *XSORT provides:*

Support to handle multiple input files up to 9.

A "Count-only" pass (SORTC) capability.

An additional specification record called S Option, identified by an S entry in column 6, which provides additional global controls including:

- Address-out file first record number
- Output file existence check
- Maximum number of records to be sorted

### *XSORT does not provide:*

Support for data types "Z" (zone) and "D" (digit) for independent use of EBCDIC zones and digits. Supports data type "C" (complete ASCII character).

Summary Sort capability.

The IBM #GSORT reserved "keywords" UDATE, UMONTH, UDAY, and UYEAR.

The "workfile data verification" feature as controlled by IBM \$DSORT and #GSORT through Header Column 34.

"Null Output Message" option as controlled by IBM #GSORT Header Column 36.

A 96 column specification form. XSORT's specification form is 80 columns. Users of \$DSORT or #GSORT using Alternate Collating Sequences of more than 80 columns will have to adapt them to the 80-column format when converting to XSORT.



XSORT error messages distinguish between two types of errors — syntax errors and execution errors.

In the discussion of these messages below, the syntactical errors are covered first. They are broken down by the category under which they can be grouped. General syntactical errors are followed by those which occur under the headings of the various types of specifications: Header, ALTSEQ, S Option, Record Type and Field Description.

Syntax messages are caused by the detection of some type of inconsistency during the interpretation of the XSORT specifications. They are included in the listings of the specifications on \$STD-LIST and usually follow immediately the specification section to which they belong — Header, ALTSEQ, S Option, Record Type, or Field Description.

The error number associated with a message is printed on the right hand margin. When an error can be associated with a particular set of columns on a specification line, that set of columns is underlined with asterisks (\*\*\*\*\*).

On the following pages is a list of errors with their corresponding numbers and further explanations of their meanings when necessary.

## Error Messages

### General Errors

NUMBER	CAUSE	FURTHER EXPLANATION
202	INVALID SPECIFICATION TYPE-COL 6 MUST BE H, I, O, OR F	Only exceptions are ALTSEQ records.
1001	MORE THAN ONE HEADER SPEC	XSORT specifications must begin with one, and only one, header specification.
1004	BEGIN/END COLUMN MISSING OR INVALID NUMERIC	For both Record Type and Field Description Specifications, the "Location From/To" entries in columns 9–16 must contain valid numeric values.
1005	BEGIN COL > END COL OR END COL OUTSIDE OF RECORD	For both Record Type and Field Description specifications, the "Location From" entry in columns 9–12 must not be greater than the "To" entry in columns 13–16. Also, the "To" value must not exceed the actual record length of the XSORTIN file.
1012	BEGIN COL MUST BE BLANK	"From location" (Columns 9–12) must be blank or the same as "To location" (Columns 13–16) for conditionally forced control fields.
1019	INVALID DECIMAL DIGIT	Constant in columns 20–80 must contain valid decimal digits and "+" or "-" in first position.
1024	MAX U FIELD LENGTH IS 28 DIGITS	The maximum length of any unpacked numeric field is 28 digits.

### Syntax Messages

#### Header Errors

NUMBER	CAUSE	FURTHER EXPLANATION
103	MUST BE SORTA, SORTC, OR SORTR	Columns 7–12 must contain SORTA, SORTC, or SORTR.
151	NO CONTROL FIELD LENGTH SPECIFIED	Columns 13–17 must contain the largest sum of control field lengths of any record section (in bytes).
162	INVALID NUMBER IN OUTPUT RECORD LENGTH FIELD	Columns 29–32 must contain a valid numerical value.
1002	BAD KEY RETENTION OPT -- MUST BE BLANK OR X	Column 28 must contain either a blank, to retain control fields or output records, or "X" to drop the control fields.
1003	ASCENDING/DESCENDING OPT INVALID-COL 18 MUST BE A OR D	
1009	ASTSEQ MUST BE S OR BLANK -COL 26	

## Alternate Collating Sequence Errors

NUMBER	CAUSE	FURTHER EXPLANATION
1015	ALTSEQ TABLE MUST END WITH **	ALTSEQ specifications must be terminated by a record containing "****" in columns 1-2.
1016	INVALID HEX DIGIT	Each hexadecimal character must be in the range of ASCII characters "0" through "F".
1023	MAX FIELD LENGTH WITH ALTSEQ IS 256 CHARACTERS	When an alternate collating sequence is being used, the maximum length of any control or data field is 256 characters.
1025	PRINT OPTION (COL 27) MUST BE 0, 1, 2, OR 3	



## S Option Errors

NUMBER	CAUSE	FURTHER EXPLANATION
1020	MAXIMUM NUMBER OF RECS TO BE SORTED INVALID	Columns 10-16 must be blank or contain a valid, non-negative numeric value.
1021	INVALID EXISTENCE OPTION - COL 8 MUST BE Y, N, OR BLANK	
1022	INVALID FIRST RECORDS # - COL 7 MUST BE 0, 1, OR BLANK	

## Record Type Errors

NUMBER	CAUSE	FURTHER EXPLANATION
206	INVALID CONTINUATION-COL 7 MUST BE A, 0, OR BLANK	
210	INVALID DATA TYPE-COL 8 MUST BE A, C, P OR U	
212	INVALID SET-OMIT & FIELD DESC BUT NO INCLUDE	Omit sets must always be followed by at least one Include prior to entry of Field Description specs.
214	INCLUDE OR OMIT SPECS AFTER AN INCLUDE ALL	"Include all," if used, must be specified after all Include/Omit sets for this record section.
216	INCOMPLETE SET-INCLUDE OR OMIT W/OUT FOLLOWING FIELD DESC'S	Include/Omit sets must be followed by Field Description specs.
236	INVALID RELATION-COL 17-18 MUST BE EQ, NE, LT, GT, LE, OR GE	
238	INVALID FACTOR 2 TYPE-COL 19 MUST BE C OR F	

## Error Messages

NUMBER	CAUSE	FURTHER EXPLANATION
1006	FIELD LENGTHS IN COMPARE ARE DIFFERENT	For a field compare ("F" in column 19), Factor 1 (columns 9–16) field length must be the same as Factor 2 (columns 20-17) field length.
1018	PACKED CONSTANT MUST HAVE + OR - SIGN	Constant in columns 20–80 must include a trailing "+" or "-" sign when being compared to a packed data field.

## Field Description Errors

NUMBER	CAUSE	FURTHER EXPLANATION
246	INVALID FIELD TYPE -COL 7 MUST BE N, O, D, F OR *	
252	CONTROL FIELD LENGTH EXCEEDS HEADER VALUE	The sum of control field lengths for this record section exceeds the maximum sum specified in Header columns 13–17.
262	DATA LENGTH EXCEEDS HEADER VALUE	Total length of data for the output record exceeds the length specified in Header columns 29–32.
1007	DATA TYPE INVALID -COL 8 MUST BE P, U, C, OR V	
1008	RECORD CHAR ONLY LEGAL IN CONDITIONAL FORCE	Record character (column 17) is only used for specification of a conditionally forced control field.
1010	FORCE ALL WITHOUT CONDITIONAL FORCE PRECEDING	A force-all control field must be preceded by at least one conditionally forced control field.
1011	INVALID CONDITIONAL FORCE CONTINUATION	Continuation (column 19) is not valid with: *an unconditionally forced control field *a conditionally forced control field that immediately follows a one-character normal or opposite control field that is not at the same record location.
1013	FORCE ALL & CONTROL FIELD CONTINUE MUTUALLY EXCLUSIVE	A set of conditionally forced control fields that immediately follow a normal or opposite control field cannot be followed by a force-all control field.
1014	DATA FIELDS NOT ALLOWED WITH ADDRESS-OUT SORT	Data fields ("D" in column 7) cannot be specified with an Address-out sort (SORTA in Header columns 7–11).

## Unnumbered Errors

The two unnumbered errors belong in the syntax error group.

CAUSE	FURTHER EXPLANATION
NO HEADER SPEC	XSORT specifications must begin with one and only one header specification.
NO FIELD DESC SPEC	Field description specifications (formats) must be included for all types of sorts (SORTA, SORTR, and SORTC).



## Error Messages

### Execution Messages

Execution messages are caused by some abnormal condition occurring during the actual sort, after the specifications have been analyzed. When one of these errors occurs, the following takes place:

- a "SORTSTONE" is printed showing one of the error messages printed below
- a "File Information Display" is printed if the error involves a file
- the sort is aborted. The error number, always preceded by a - (minus sign), appears after the PARAM = in the abort message

NUMBER	CAUSE	FURTHER EXPLANATION
-1	ERROR ON XSORTTEXT	Error on opening or reading XSORTTEXT file.
-2	ERROR ON \$STDLIST	Error on print to \$STDLIST.
-3	ERROR ON XSORTIN	Error on opening, accessing, or closing XSORTIN file.
-4	ERROR ON XSORTOUT	Error on opening, accessing, or closing XSORTOUT file or XSORT999 file (used for inplace sort).
-5	XSORTOUT EXISTS BUT NOT XSORTIN	If the XSORTOUT file already exists, it must be the same as XSORTIN. (This restriction can be overridden by specifying = "Y" in S Option column 8).
-6	XSORTIN RECORD BIGGER THAN BUFFER-CHANGE MAX'IN'REC	Record length of XSORTIN file exceeds 1000 characters. MAX'IN'REC is a value hard-coded in XSORT source code and so can only be changed by recompilation of XSORT by HP.
-10	TERMINAL ERROR IN SPECS-SEE LISTING	One of several errors occurred in the specifications. See the listings for the specific message.
-11	OUT OF TABLE SPACE-IBUF TOO SMALL	Not enough room in the internal routine table for all Record and Field specifications. The number of specifications must be reduced to fit within the 8192 character internal table. The amount of space (in characters) required in the table can be calculated by the following formula:  $30 + 12 * (\text{number of Record and Field Specs})$ $+ (\text{sum of lengths of constants on Record Specs})$ $+ 512 (\text{if ALTSEQ is used})$
-12	SORTINITAL FAILED	Internal error: use of SORTINITAL intrinsic failed.
-13	SORTINPUT FAILED	Internal error: use of SORTINPUT intrinsic failed.
-14	SORTOUTPUT FAILED	Internal error: use of SORTOUTPUT intrinsic failed.
-15	ATTEMPTED TO OUTPUT AN RSAM FILE	Inplace sort not allowed when XSORTIN is an RSAM file.
-16	XSORTOUT RECORD SIZE TOO SMALL	Required output record size (computed from specifications) exceeds the actual record size of XSORTOUT.

Error Messages

NUMBER	CAUSE	FURTHER EXPLANATION
-17	WORK BUFFER TOO SMALL CHANGE MAX 'WORK' REC	Work record (control fields and data fields) cannot exceed 1000 characters. MAX'WORK'REC is a value hardcoded in XSORT source code and so can only be changed by modification and recompilation of XSORT by HP.
-18	LOADPROC OF FREADC FAILED	Internal error: loading procedure for FREADC intrinsic failed.
-19	MULTI INPUT (XSORTIN1) AND SORTA INVALID	Address-out sort (SORTA in Header columns 7-11) not allowed when multiple input files (XSORTIN1 . . . . XSORTIN9) are used.
-20	MULTI INPUT AND INPLACE SORT INVALID	Inplace sort not allowed when multiple input files (XSORTIN1. . . . XSORTIN9) are used.
-21	CAN'T SORT KSAM FILE INPLACE	Inplace sort not allowed when XSORTIN is a KSAM file.



### Collating Sequence Table

The Collating Sequence Table brings together comparative information concerning data used in sort jobs. It includes the ASCII and EBCDIC collating sequences in decimal, octal, hexadecimal and binary representations. It gives the character (graphic) or control equivalents for both ASCII and EBCDIC.

The relative position of alphabetic and numeric characters in the sorting sequence is shown in the right hand columns. This information will be helpful to those users who have experience with systems based on the EBCDIC collating sequence. A major difference between the two systems is the relative position of alphabetic and numeric characters in the collating sequence. When alphabetic characters have a higher place in the collating sequence than numeric characters, this can effect compare and matching field operations. Using Alternate Collating Sequence, it is possible to reverse this if the user prefers EBCDIC sequence.

COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
0	000	00	0000	0000	NUL	NUL		
1	001	01	0000	0001	SOH	SOH		
2	002	02	0000	0010	STX	STX		
3	003	03	0000	0011	ETX	ETX		
4	004	04	0000	0100	EOT	PF		
5	005	05	0000	0101	ENQ	HT		
6	006	06	0000	0110	ACK	LC		
7	007	07	0000	0111	BEL	DEL		
8	010	08	0000	1000	BS			
9	011	09	0000	1001	HT			
10	012	0A	0000	1010	LF	SMM		
11	013	0B	0000	1011	VT	VT		
12	014	0C	0000	1100	FF	FF		
13	015	0D	0000	1101	CR	CR		
14	016	0E	0000	1110	SO	SO		
15	017	0F	0000	1111	SI	SI		
16	020	10	0001	0000	DLE	DLE		
17	021	11	0001	0001	DC1	DC1		
18	022	12	0001	0010	DC2	DC2		
19	023	13	0001	0011	DC3	TM		
20	024	14	0001	0100	DC4	RES		
21	025	15	0001	0101	NAK	NL		
22	026	16	0001	0110	SYN	BS		
23	027	17	0001	0111	ETB	IL		
24	030	18	0001	1000	CAN	CAN		
25	031	19	0001	1001	EM	EM		
26	032	1A	0001	1010	SUB	CC		
27	033	1B	0001	1011	ESC	CU1		
28	034	1C	0001	1100	FS	IFS		
29	035	1D	0001	1101	GS	IGS		
30	036	1E	0001	1110	RS	IRS		
31	037	1F	0001	1111	US	IUS		
32	040	20	0010	0000	SP	DS		
33	041	21	0010	0001	!	SOS		
34	042	22	0010	0010	"	FS		
35	043	23	0010	0011	#			
36	044	24	0010	0100	\$	BYP		
37	045	25	0010	0101	%	LF		
38	046	26	0010	0110	&	ETB		
39	047	27	0010	0111	'	ESC		
40	050	28	0010	1000	(			
41	051	29	0010	1001	)			
42	052	2A	0010	1010	*	SM		
43	053	2B	0010	1011	+	CU2		
44	054	2C	0010	1100	,			
45	055	2D	0010	1101	-	ENQ		
46	056	2E	0010	1110	.	ACK		
47	057	2F	0010	1111	/	BEL		

COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
48	060	30	0011	0000	0		0	
49	061	31	0011	0001	1		1	
50	062	32	0011	0010	2	SYN	2	
51	063	33	0011	0011	3		3	
52	064	34	0011	0100	4	PN	4	
53	065	35	0011	0101	5	RS	5	
54	066	36	0011	0110	6	UC	6	
55	067	37	0011	0111	7	EOT	7	
56	070	38	0011	1000	8		8	
57	071	39	0011	1001	9		9	
58	072	3A	0011	1010	:			
59	073	3B	0011	1011	;	CU3		
60	074	3C	0011	1100	<	DC4		
61	075	3D	0011	1101	=	NAK		
62	076	3E	0011	1110	>			
63	077	3F	0011	1111	?	SUB		
64	100	40	0100	0000	@	SP		
65	101	41	0100	0001	A		+1	
66	102	42	0100	0010	B		+2	
67	103	43	0100	0011	C		+3	
68	104	44	0100	0100	D		+4	
69	105	45	0100	0101	E		+5	
70	106	46	0100	0110	F		+6	
71	107	47	0100	0111	G		+7	
72	110	48	0100	1000	H		+8	
73	111	49	0100	1001	I		+9	
74	112	4A	0100	1010	J	¢	-1	
75	113	4B	0100	1011	K	.	-2	
76	114	4C	0100	1100	L	<	-3	
77	115	4D	0100	1101	M	(	-4	
78	116	4E	0100	1110	N	+	-5	
79	117	4F	0100	1111	O		-6	
80	120	50	0101	0000	P	&	-7	
81	121	51	0101	0001	Q		-8	
82	122	52	0101	0010	R		-9	
83	123	53	0101	0011	S			
84	124	54	0101	0100	T			
85	125	55	0101	0101	U			
86	126	56	0101	0110	V			
87	127	57	0101	0111	W			
88	130	58	0101	1000	X			
89	131	59	0101	1001	Y			
90	132	5A	0101	1010	Z	!		
91	133	5B	0101	1011	[	\$		
92	134	5C	0101	1100	\	*		
93	135	5D	0101	1101	]	)		
94	136	5E	0101	1110	^	;		
95	137	5F	0101	1111	_			

COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
96	140	60	0110	0000	.	-		
97	141	61	0110	0001	a	/		
98	142	62	0110	0010	b			
99	143	63	0110	0011	c			
100	144	64	0110	0100	d			
101	145	65	0110	0101	e			
102	146	66	0110	0110	f			
103	147	67	0110	0111	g			
104	150	68	0110	1000	h			
105	151	69	0110	1001	i			
106	152	6A	0110	1010	j			
107	153	6B	0110	1011	k	,		
108	154	6C	0110	1100	l	%		
109	155	6D	0110	1101	m	~		
110	156	6E	0110	1110	n	>		
111	157	6F	0110	1111	o	?		
112	160	70	0111	0000	p			
113	161	71	0111	0001	q			
114	162	72	0111	0010	r			
115	163	73	0111	0011	s			
116	164	74	0111	0100	t			
117	165	75	0111	0101	u			
118	166	76	0111	0110	v			
119	167	77	0111	0111	w			
120	170	78	0111	1000	x			
121	171	79	0111	1001	y	,		
122	172	7A	0111	1010	z	:		
123	173	7B	0111	1011	{	#	+0	
124	174	7C	0111	1100		@		
125	175	7D	0111	1101	}	,	-0	
126	176	7E	0111	1110	~	=		
127	177	7F	0111	1111	DEL	"		
128	200	80	1000	0000		a		
129	201	81	1000	0001		b		
130	202	82	1000	0010		c		
131	203	83	1000	0011		d		
132	204	84	1000	0100		e		
133	205	85	1000	0101		f		
134	206	86	1000	0110		g		
135	207	87	1000	0111		h		
136	210	88	1000	1000		i		
137	211	89	1000	1001				
138	212	8A	1000	1010				
139	213	8B	1000	1011				
140	214	8C	1000	1100				
141	215	8D	1000	1101				
142	216	8E	1000	1110				
143	217	8F	1000	1111				

COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
144	220	90	1001	0000				
145	221	91	1001	0001		j		
146	222	92	1001	0010		k		
147	223	93	1001	0011		l		
148	224	94	1001	0100		m		
149	225	95	1001	0101		n		
150	226	96	1001	0110		o		
151	227	97	1001	0111		p		
152	230	98	1001	1000		q		
153	231	99	1001	1001		r		
154	232	9A	1001	1010				
155	233	9B	1001	1011				
156	234	9C	1001	1100				
157	235	9D	1001	1101				
158	236	9E	1001	1110				
159	237	9F	1001	1111				
160	240	A0	1010	0000				
161	241	A1	1010	0001		~		
162	242	A2	1010	0010		s		
163	243	A3	1010	0011		t		
164	244	A4	1010	0100		u		
165	245	A5	1010	0101		v		
166	246	A6	1010	0110		w		
167	247	A7	1010	0111		x		
168	250	A8	1010	1000		y		
169	251	A9	1010	1001		z		
170	252	AA	1010	1010				
171	253	AB	1010	1011				
172	254	AC	1010	1100				
173	255	AD	1010	1101				
174	256	AE	1010	1110				
175	257	AF	1010	1111				
176	260	B0	1011	0000				
177	261	B1	1011	0001				
178	262	B2	1011	0010				
179	263	B3	1011	0011				
180	264	B4	1011	0100				
181	265	B5	1011	0101				
182	266	B6	1011	0110				
183	267	B7	1011	0111				
184	270	B8	1011	1000				
185	271	B9	1011	1001				
186	272	BA	1011	1010				
187	273	BB	1011	1011				
188	274	BC	1011	1100				
189	275	BD	1011	1101				
190	276	BE	1011	1110				
191	277	BF	1011	1111				



COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
192	300	C0	1100	0000		{		+0
193	301	C1	1100	0001		A		+1
194	302	C2	1100	0010		B		+2
195	303	C3	1100	0011		C		+3
196	304	C4	1100	0100		D		+4
197	305	C5	1100	0101		E		+5
198	306	C6	1100	0110		F		+6
199	307	C7	1100	0111		G		+7
200	310	C8	1100	1000		H		+8
201	311	C9	1100	1001		I		+9
202	312	CA	1100	1010				
203	313	CB	1100	1011				
204	314	CC	1100	1100		]		
205	315	CD	1100	1101				
206	316	CE	1100	1110		}		
207	317	CF	1100	1111				
208	320	D0	1101	0000		}		-0
209	321	D1	1101	0001		J		-1
210	322	D2	1101	0010		K		-2
211	323	D3	1101	0011		L		-3
212	324	D4	1101	0100		M		-4
213	325	D5	1101	0101		N		-5
214	326	D6	1101	0110		O		-6
215	327	D7	1101	0111		P		-7
216	330	D8	1101	1000		Q		-8
217	331	D9	1101	1001		R		-9
218	332	DA	1101	1010				
219	333	DB	1101	1011				
220	334	DC	1101	1100				
221	335	DD	1101	1101				
222	336	DE	1101	1110				
223	337	DF	1101	1111				
224	340	E0	1110	0000		\		
225	341	E1	1110	0001				
226	342	E2	1110	0010		S		
227	343	E3	1110	0011		T		
228	344	E4	1110	0100		U		
229	345	E5	1110	0101		V		
230	346	E6	1110	0110		W		
231	347	E7	1110	0111		X		
232	350	E8	1110	1000		Y		
233	351	E9	1110	1001		Z		
234	352	EA	1110	1010				
235	353	EB	1110	1011				
236	354	EC	1110	1100				
237	355	ED	1110	1101				
238	356	EE	1110	1110				
239	357	EF	1110	1111				

Data

COLLATING SEQUENCE				CHARACTER/ CONTROL		NUMERIC CHARACTER		
Dec	Oct	Hex	Binary		ASCII	EBCDIC	ASCII	EBCDIC
240	360	F0	1111	0000		0		0
241	361	F1	1111	0001		1		1
242	362	F2	1111	0010		2		2
243	363	F3	1111	0011		3		3
244	364	F4	1111	0100		4		4
245	365	F5	1111	0101		5		5
246	366	F6	1111	0110		6		6
247	367	F7	1111	0111		7		7
248	370	F8	1111	1000		8		8
249	371	F9	1111	1001		9		9
250	372	FA	1111	1010				
251	373	FB	1111	1011				
252	374	FC	1111	1100				
253	375	FD	1111	1101				
254	376	FE	1111	1110				
255	377	FF	1111	1111				

XSORT ALTERNATE COLLATING SEQUENCE CODING WORKSHEET

Code	Character	Hexa decimal Entry	Octal Entry	Replaces/ Replaced By
00000000	NUL	00	000	
00000001	SOH	01	001	
00000010	STX	02	002	
00000011	ETX	03	003	
00000100	EOT	04	004	
00000101	ENO	05	005	
00000110	ACK	06	006	
00000111	BEL	07	007	
00001000	BS	08	010	
00001001	HT	09	011	
00001010	LF	0A	012	
00001011	VT	0B	013	
00001100	FF	0C	014	
00001101	CR	0D	015	
00001110	SO	0E	016	
00001111	SI	0F	017	
00010000	DLE	10	020	
00010001	DC1	11	021	
00010010	DC2	12	022	
00010011	DC3	13	023	
00010100	DC4	14	024	
00010101	NAK	15	025	
00010110	SYN	16	026	
00010111	ETB	17	027	
00011000	CAN	18	030	
00011001	EM	19	031	
00011010	SUB	1A	032	
00011011	ESC	1B	033	
00011100	FS	1C	034	
00011101	GS	1D	035	
00011110	RS	1E	036	
00011111	US	1F	037	
00100000	SP	20	040	
00100001	...	21	041	
00100010	...	22	042	
00100011	...	23	043	
00100100	...	24	044	
00100101	...	25	045	
00100110	...	26	046	
00100111	...	27	047	
00101000	...	28	050	
00101001	...	29	051	
00101010	...	2A	052	
00101011	...	2B	053	
00101100	...	2C	054	
00101101	...	2D	055	
00101110	...	2E	056	
00101111	...	2F	057	
00110000	...	30	060	
00110001	...	31	061	
00110010	...	32	062	
00110011	...	33	063	
00110100	...	34	064	
00110101	...	35	065	
00110110	...	36	066	
00110111	...	37	067	
00111000	...	38	070	
00111001	...	39	071	
00111010	...	3A	072	
00111011	...	3B	073	
00111100	...	3C	074	
00111101	...	3D	075	
00111110	...	3E	076	
00111111	...	3F	077	

Code	Character	Hexa decimal Entry	Octal Entry	Replaces/ Replaced By
01004000	@	40	100	
01000001	A	41	101	
01000010	B	42	102	
01000011	C	43	103	
01000100	D	44	104	
01000101	E	45	105	
01000110	F	46	106	
01000111	G	47	107	
01001000	H	48	110	
01001001	I	49	111	
01001010	J	4A	112	
01001011	K	4B	113	
01001100	L	4C	114	
01001101	M	4D	115	
01001110	N	4E	116	
01001111	O	4F	117	
01010000	P	50	120	
01010001	Q	51	121	
01010010	R	52	122	
01010011	S	53	123	
01010100	T	54	124	
01010101	U	55	125	
01010110	V	56	126	
01010111	W	57	127	
01011000	X	58	130	
01011001	Y	59	131	
01011010	Z	5A	132	
01011011	[	5B	133	
01011100	\	5C	134	
01011101	]	5D	135	
01011110	^	5E	136	
01011111	_	5F	137	
01100000	...	60	140	
01100001	...	61	141	
01100010	...	62	142	
01100011	...	63	143	
01100100	...	64	144	
01100101	...	65	145	
01100110	...	66	146	
01100111	...	67	147	
01101000	...	68	150	
01101001	...	69	151	
01101010	...	6A	152	
01101011	...	6B	153	
01101100	...	6C	154	
01101101	...	6D	155	
01101110	...	6E	156	
01101111	...	6F	157	
01110000	...	70	160	
01110001	...	71	161	
01110010	...	72	162	
01110011	...	73	163	
01110100	...	74	164	
01110101	...	75	165	
01110110	...	76	166	
01110111	...	77	167	
01111000	...	78	170	
01111001	...	79	171	
01111010	...	7A	172	
01111011	...	7B	173	
01111100	...	7C	174	
01111101	...	7D	175	
01111110	...	7E	176	
01111111	DEL	7F	177	

Code	Character	Hexa decimal Entry	Octal Entry	Replaces/ Replaced By
10000000	K0	80	200	
10000001	K1	81	201	
10000010	K2	82	202	
10000011	K3	83	203	
10000100	K4	84	204	
10000101	K5	85	205	
10000110	K6	86	206	
10000111	K7	87	207	
10001000	K8	88	210	
10001001	K9	89	211	
10001010	K10	8A	212	
10001011	K11	8B	213	
10001100	K12	8C	214	
10001101	K13	8D	215	
10001110	K14	8E	216	
10001111	K15	8F	217	
10010000	K16	90	220	
10010001	K17	91	221	
10010010	K18	92	222	
10010011	K19	93	223	
10010100	K20	94	224	
10010101	K21	95	225	
10010110	K22	96	226	
10010111	K23	97	227	
10011000	K24	98	230	
10011001	K25	99	231	
10011010	K26	9A	232	
10011011	K27	9B	233	
10011100	K28	9C	234	
10011101	K29	9D	235	
10011110	K30	9E	236	
10011111	K31	9F	237	
10100000	N0	A0	240	
10100001	N1	A1	241	
10100010	N2	A2	242	
10100011	N3	A3	243	
10100100	N4	A4	244	
10100101	N5	A5	245	
10100110	N6	A6	246	
10100111	N7	A7	247	
10101000	N8	A8	250	
10101001	N9	A9	251	
10101010	N10	AA	252	
10101011	N11	AB	253	
10101100	N12	AC	254	
10101101	N13	AD	255	
10101110	N14	AE	256	
10101111	N15	AF	257	
10110000	N16	B0	260	
10110001	N17	B1	261	
10110010	N18	B2	262	
10110011	N19	B3	263	
10110100	N20	B4	264	
10110101	N21	B5	265	
10110110	N22	B6	266	
10110111	N23	B7	267	
10111000	N24	B8	270	
10111001	N25	B9	271	
10111010	N26	BA	272	
10111011	N27	BB	273	
10111100	N28	BC	274	
10111101	N29	BD	275	
10111110	N30	BE	276	
10111111	N31	BF	277	

Code	Character	Hexa decimal Entry	Octal Entry	Replaces/ Replaced By
11000000	N32	C0	300	
11000001	N33	C1	301	
11000010	N34	C2	302	
11000011	N35	C3	303	
11000100	N36	C4	304	
11000101	N37	C5	305	
11000110	N38	C6	306	
11000111	N39	C7	307	
11001000	N40	C8	310	
11001001	N41	C9	311	
11001010	N42	CA	312	
11001011	N43	CB	313	
11001100	N44	CC	314	
11001101	N45	CD	315	
11001110	N46	CE	316	
11001111	N47	CF	317	
11010000	N48	D0	320	
11010001	N49	D1	321	
11010010	N50	D2	322	
11010011	N51	D3	323	
11010100	N52	D4	324	
11010101	N53	D5	325	
11010110	N54	D6	326	
11010111	N55	D7	327	
11011000	N56	D8	330	
11011001	N57	D9	331	
11011010	N58	DA	332	
11011011	N59	DB	333	
11011100	N60	DC	334	
11011101	N61	DD	335	
11011110	N62	DE	336	
11011111	N63	DF	337	
11100000	G0	E0	340	
11100001	G1	E1	341	
11100010	G2	E2	342	
11100011	G3	E3	343	
11100100	G4	E4	344	
11100101	G5	E5	345	
11100110	G6	E6	346	
11100111	G7	E7	347	
11101000	G8	E8	350	
11101001	G9	E9	351	
11101010	G10	EA	352	
11101011	G11	EB	353	
11101100	G12	EC	354	
11101101	G13	ED	355	
11101110	G14	EE	356	
11101111	G15	EF	357	
11110000	G16	F0	360	
11110001	G17	F1	361	
11110010	G18	F2	362	
11110011	G19	F3	363	
11110100	G20	F4	364	
11110101	G21	F5	365	
11110110	G22	F6	366	
11110111	G23	F7	367	
11111000	G24	F8	370	
11111001	G25	F9	371	
11111010	G26	FA	372	
11111011	G27	FB	373	
11111100	G28	FC	374	
11111101	G29	FD	375	
11111110	G30	FE	376	
11111111	E0	FF	377	



# **XSORT SPECIFICATION FORMS**

**APPENDIX**

**D**

The forms for specifications are for your convenience. Copies may be made for use in your sort jobs. Hewlett-Packard holds the copyright for the forms and asks that they be copied only for your particular applications.



HEADER

Line Number	Spec. Type (H)	Type of Sort	Largest sum of control field lengths of any record section (in bytes)	Sequence (A or D)	Reserved	Alternate Coll. Seq. (S)	Print Option	Output Option (X)	Length of output record	Reserved	Comments (any characters)																																																												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

Entries can extend to 80 columns →

→ Insert AL TSEQ specifications here (if used).  
 → Insert S OPTION specification here (if used).

RECORD TYPE

Line Number	Spec. Type (I or O)	Continuation (A, O, or *)	C, P, or U	Factor 1	Rel.	Factor 2 (Field or Constant)	Comments (any characters)																																																																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

EQ, NE, LT, GT, LE, GE, T, C, or U

Location from to

Constant (any ASCII characters)

Location from to

Record Name

Entries can extend to 80 columns →

FIELD DESCRIPTION

Line Number	Spec. Type (F)	Type (N, O, F, D, or *)	P, U, C, or V	Location	Record character	Substitute character	Continuation	Comments (any characters)																																																															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

Location from to

Record character

Substitute character

Continuation

Field Name

Entries can extend to 80 columns →









\$STDIN, use of designator, 1-6, 1-10, 7-2  
 \$STDLIST, use of designator, 1-6, 7-2, B-1  
 \*\* (double asterisk) — use with ALTSEQ, 3-2

## A

Address-out first file record number (S option, column 7), 4-2  
 Address-out sort (SORTA)  
   definition, 1-5  
   first file record number, 4-2  
   overview of specifications, 1-13  
   output, 1-5  
   results in application, 8-12  
 Alternate collating sequence coding and worksheet, C-9  
 Alternate collating sequence (ALTSEQ)  
   cannot be used with packed data, 2-5  
   effects on other coding, 3-3  
   effects on program, 3-3  
   number of columns which may be used, 3-1  
   order of specifications, 3-3  
   rules for coding, 3-2  
   sample statements, 3-4  
   with header column 26, 2-5  
   use with include sets, 5-6  
   use with omit Sets, 5-7  
 AND lines  
   definition, 5-5  
   with include sets, 5-6  
   with omit sets, 5-7  
 Applications  
   #1 Mixed include and omit set in one section, 8-1  
   #2 Multiple sections and sets, 8-2  
   #3, Sort 1 (Record-out), 8-9  
   #3, Sort 2 (Address-out), 8-13  
   #3, Sort 3 (Record-out, multiple file), 8-20  
 Ascending or descending sequence (header specs, column 18), 2-5  
 ASCII collating sequence, C-2  
 ASCII used as standard collating sequence, 3-1

## C

C/P/U data types (record type specifications, column 8), 5-8  
 C,P,U or V data types (field description specs, column 8), 6-9  
 Changing collating sequence (header specs, column 26), 2-5  
 Changing collating sequence (using ALTSEQ), 3-1  
 Character (C), definition, 5-8  
 Character data, maximum field length, 5-8  
 Collating sequence (header specs, column 26), 2-5  
 Collating sequence table, C-3  
 Column 1 to 5 (field description specs) — line number, 6-3  
 Column 1 to 5 (header specs) — line number, 2-3

Column 1 to 5 (record type specs) — line number, 5-3  
 Column 1 to 5 (S option) — line number, 4-2  
 Column 6 (field description specs) — line type, 6-3  
 Column 6 (header specs) — line type, 2-3  
 Column 6 (record type specs) — spec type, 5-3  
 Column 6 (S option) — S entry, 4-2  
 Column 7 (field description specs) — field type or comments, 6-3  
 Column 7 (record types specs) — continuation or comments, 5-5  
 Column 7 (S option) — Address-out first file record number, 4-2  
 Column 7 to 12 (header specs) — type of sort, 2-3  
 Column 8 (field description specs) — data type (C,P,U, or V), 6-9  
 Column 8 (record type specs) — data type (C/P/U), 5-8  
 Column 8 (S option) — inplace sort override, 4-3  
 Column 9 to 16 (field description specs) — field locations, 6-10  
 Column 9 to 16 (record type specs) — factor 1 field length, 5-9  
 Column 10 to 16 (S option) — maximum number of records to be sorted, 4-3  
 Column 13 to 17 (header specs) — largest sum of control field length of any record section (in bytes), 2-3  
 Column 17 (field description specs) — conditionally forced characters, 6-11  
 Column 17 to 18 (record type specs) — type of comparison, 5-10  
 Column 18 (field description specs) — forced character, 6-12  
 Column 18 (header specs) — ascending or descending sequence, 2-5  
 Column 19 (field description specs) — continuation, 6-12  
 Column 19 (record type specs) — field or constant, 5-11  
 Column 20 to 27 (record type specs) — factor 2 field, 5-11  
 Column 20 to 80 (record type specs) — factor 2 constant, 5-11  
 Column 26 (header specs) — collating sequence, 2-5  
 Column 27 (header specs) — print option, 2-5  
 Column 28 (header specs) — output option for record-out sort, 2-6  
 Column 29 to 32 (header specs) — output record length for record-out sorts, 2-6  
 Column 40 to 80 (field description specs) — comments, 6-22  
 Column 40 to 80 (header specs) — comments, 2-6  
 Column 40 to 80 (record type specs) — comments, 5-13

# INDEX (continued)

## Commands, example

- #1 job stream with specs imbedded; single input to different output, 7-6
- #2 job stream with XSORTTEXT; single input to different output, 7-9
- #3 session with specs entered interactively; single input to different output, 7-10
- #4 session using XSORTTEXT, single input to different output; case #1, 7-11
- #5 session using XSORTTEXT, input from \$STDIN and output to \$STDLIST; case #2, 7-12
- #6 job stream with specifications imbedded; single file, inplace sort, 7-13
- #7 job stream with specifications imbedded; single file to different output that already exists, 7-14
- #8 job stream with specifications imbedded; multiple file input, 7-15
- #9 SORTC (Count-only sort), 7-16
  - explanation of specifications, 7-5
  - specifications, 7-4

Commands for SORT3, 1-9, A-1

Commands grid, MPE, 7-2

Comments (field description specs, columns 40-80), 6-22

Comments (header specs, columns 40-80), 2-6

Comments (record type specs, columns 40-80), 5-13

Comparing factor 1 and factor 2, 5-10

Comparisons, type and entries, columns 17-18, record type specs, 5-10

Comparisons, using control fields, 6-3

Compatibility of XSORT with SORT3, A-1

Computing control field lengths, 2-3

Conditional force

on normal or opposite control field, 6-6

stand-alone conditional force, 6-7

using force-all, 6-7

Conditionally forced characters (field description specs, column 17), 6-11

Constant, factor 1 (see factor 1 constant)

Constant, factor 2 (see factor 2 constant)

Constants

comparing factor 1 and factor 2, 5-8

numeric, packed or unpacked, 5-12

packed, 5-13

unpacked, 5-12

Continuation (field description specs, column 19), 6-12

Continuation or comments (record type specs, column 7), 5-5

Control fields (N, O, or F) — field description specs, 6-3

Control fields

computing control field length, 2-3

definition, 1-2

dropping (see header specs, column 28), 2-6

forced control fields, 6-6

normal and opposite control fields, 6-3

opposite control fields, 6-4

packed, normal or opposite, 6-9

reasons for dropping, 2-6

sequencing information, 6-4

Count-only sort (SORTC)

definition, 1-5

overview of specifications, 1-14

use in command example, 7-16

## D

Data

character, maximum field length, 6-9

collating sequence table (ASCII and EBCDIC), C-2

forcing a data character into record, 6-9

maximum input record size, 1-11

packed decimal, 6-9

type (C/P/U) with record type specs, 5-8

type C,P,U,V with field description specifications, 6-9

unpacked decimal, 6-9

used in work records (C/P/U/V), 6-9

Data fields

in column 7, field description specs, 6-8

maximum length for C/P/U, 5-8

maximum length for C/P/U, 5-8

use in comparisons, 5-8

## E

EBCDIC collating sequence, C-2

Error messages

discussion, B-1

execution messages, B-5

general errors, B-2

syntax messages, B-2

types, B-1

unnumbered errors, B-4

on \$STDLIST, 1-6

## F

Factor 1 field length (record type specs, columns 9-16), 5-9

Factor 1 constant

coding rules, 5-10

definition, 5-9

explanation, 5-9

maximum length, 5-9

Factor 2 constant (record type specs, columns 20-80), 5-11

Factor 2

coding, 5-11

field or constant, 5-11

maximum length, 5-11

Factor 2 field (record type specs, columns 20-27), 5-11

Field description specifications  
discussion, 6-1  
overview of column entries, 6-2  
column entries, 6-3  
column 6, line type, 6-3  
column 7, field type or comments, 6-3  
column 8, data type (C,P,U or V), 6-9  
columns 9-16, field locations, 6-10  
column 17, conditionally forced characters, 6-11  
column 18, forced character, 6-12  
column 19, continuation, 6-12  
columns 40-80, comments, 6-22  
Field in comparison of factor 1 and factor 2, 5-11  
Field length in columns 9-16, field description specs, 6-11  
Field locations (field description specs, columns 9-16), 6-10  
Field or constant (record type specs, column 19), 5-11  
Field type or comments (field description specs, column 7), 6-3  
Files (see input files, multiple input files, output files)  
Force-all, 6-7  
Force character, summary, 6-13  
Forced character (field description specs, column 18), 6-12  
Forced control characters  
example of conditional force on normal or opposite control field, 6-14  
example of force-all, 6-20  
example of stand-alone conditional force, 6-16  
example of unconditionally forced character, 6-14  
From entry, columns 9-12, field description specifications, indicating start of record positions for a field, 6-10  
From entry, columns 9-12, record type specifications, indicating where factor 1 field begins, 5-9  
From entry, columns 20-23, record type specifications, indicating where factor 2 field begins, 5-11

## H

Header specifications  
discussion, 2-1  
overview of column entries, 2-2  
column entries, 2-3  
column 1 to 5, line number, 2-3  
column 6, line type, 2-3  
column 7 to 12, type of sort, 2-3  
column 13 to 17, largest sum of control field lengths of any record section in bytes, 2-3  
column 18, ascending or descending sequence, 2-5  
column 26, collating sequence, 2-5  
column 27, print option, 2-5  
column 28, output option for record-out sort, 2-6  
column 40 to 80, job description comments, 2-6  
column 29-32, output record length for record-out sorts, 2-6  
How XSORT works, 1-2

## I

IBM System/3 \$DSORT and IBM System 32 & 34 #GSORT, compared to XSORT, A-3  
IBM System/3 \$DSORT and System/34 #GSORT, relationship to XSORT, 1-1  
IMAGE data base file, not processed by XSORT, 1-10  
Implied include-all, definition, 5-4  
Importance of column 8 entry, record type specs, 5-8  
Include set, definition, 5-4  
Include sets, relating columns 7 to 8 in record type specs, 5-6  
Include-all, definition, 5-4  
Inplace sort, commands required, 7-13  
Inplace sort, discussion with file designators, 1-11  
Inplace sort override (S option, column 8), 4-4  
Input files  
KSAM, 1-6  
MPE, 1-6, 7-1  
multiple file designators, 1-9  
names, 1-9  
non-disc, 1-6  
rearrangement by XSORT, 1-2  
rules for use of multiple file input, 1-10  
single file designator, 1-9  
types accepted by XSORT, 1-6  
when KSAM files are used, 1-10

## K

KSAM files  
first record number 1-10  
with address-out sorts, 1-10  
with deleted records, 1-10

## L

Largest sum of control fields lengths of any record section in bytes (header specs, columns 13-17), 2-2  
Length restrictions  
control fields of a record section, 2-3  
data type in record type specs, 5-8  
data type in field description specs, 6-9  
factor 1 field, 5-9  
factor 2 field, 5-11  
factor 2 constant, 5-11  
field locations, 6-11  
forced characters, 6-12  
numeric constants, packed and unpacked, 5-12  
output record length for record-out sorts, 2-6  
packed control fields, 6-9  
Line number (field description specs, column 6), 6-3  
Line number (header specs, columns 1-5), 2-3  
Line number (record type specs, columns 1-5), 5-3  
Line number (S option, columns 1-5), 4-2  
Line type (field description specs, column 6), 6-3  
Line type (header specs, column 6), 2-3

# INDEX (continued)

## M

- Maximum number of records to be sorted (S option, columns 10-16), 4-3
- Maximum number of records, when to specify, 1-10
- Mixing sets in record type specs, 5-7
- MPE commands, used with examples, 7-1
- MPE commands, what they provide in program, 1-9
- Multiple input files
  - example showing commands used, 7-15
  - example of sort, 8-19
  - number accepted, 1-1
  - special considerations, 1-10

## N

- Negative unpacked constants, coding, 5-12
- Number restrictions,
  - also see length restrictions
  - columns in spec forms, 1-8
  - data input record size, 1-11
  - for converting ALTSEQ specs, 3-1
  - input files allowed, 1-1
  - number of records that can be sorted, 4-3
- Numeric constant, packed or unpacked, 5-12
- Numeric data, packed, maximum field length, 5-8
- Numeric data, unpacked, maximum field length, 5-8

## O

- Omit set, definition, 5-4
- Omit sets, relating columns 7/8 entries in record type & specs, 5-7
- OR lines
  - definition, 5-5
  - with Include Sets, 5-6
  - use with Omit Sets, 5-7
- Order of specifications, requirements, 1-7
- Output file
  - formal file designator, 1-11
  - possible contents, 1-6
  - with inplace sort, 1-11
- Output option for record-out sort (header specs, column 28), 2-6
- Output record fields, calculating length, 2-6
- Output record length for record-out sorts (header specifications, columns 29-32), 2-6
- Overview of XSORT specifications, 1-12
- Overview of specifications
  - for Address-out (SORTA) sort, 1-13
  - for Count-only (SORTC) sort, 1-14
  - for Record-out (SORTR) sort, 1-12

## P

- Packed (P) data, definition, 5-8
- Packed data, cannot be used with ALTSEQ, 2-5
- Packed constants, 5-13
- Packed numeric constants, 5-12
- Packed numeric data, maximum field length, 5-8
- Print option (header specs, column 27), 2-5
- Print option, explanation, 2-5

## R

- Record sections
  - definition, 5-3
  - definition of an include-all, 5-4
  - definition of implied include-all, 5-4
  - in relation to sets and types (chart), 8-3
  - length calculation, 2-3
  - maximum length, 2-3
  - multiple sections and sets in application, 8-2
  - when you use more than one section, 6-3
- Record sets
  - definition, 5-4
  - definition of an omit set, 5-4
  - definition of an include set, 5-4
  - in relation to sections and types (chart), 8-3
  - mixed include/omit sets in application, 8-1
- Record type specifications
  - discussion, 5-1
  - overview of column entries, 5-2
  - column entries, 5-3
  - columns 1-5, line number, 5-3
  - column 6, spec type, 5-3
  - column 7, continuation or comments, 5-5
  - column 8, data type (C/P/U), 5-8
  - relating columns 7 & 8 entries, 5-6
  - columns 9-16, factor 1 field length, 5-9
  - columns 17-18, type of comparisons, 5-10
  - column 19, field or constant, 5-11
  - columns 20-80, factor 2 constant, 5-11
  - columns 20-27, factor 2 field, 5-11
  - columns 40-80, comments, 5-13
- Record types, definition, 5-3
- Record types, sections, and sets — chart showing relationship, 8-3
- Record-out sort (SORTR)
  - calculating record field lengths when dropping control fields, 2-6
  - calculating record field lengths when not dropping control fields, 2-6
  - data field entry, 6-8
  - definition, 1-5
  - overview of specifications, 1-12
  - use in applications, 8-6, 8-20

Relating record types, sections and sets — chart, 8-3  
Running the XSORT program (diagram), 1-6  
Running XSORT in job stream, 7-2  
Running XSORT interactively, 7-2

## S

S entry (S option, column 6), 4-2  
Sections, see record sections  
Sequence specifications — forms for copying, D-3, D-5  
Sets, see record sets  
Sort specifications  
    chart showing order of entry, 1-8  
    forms for copying, D-1  
    need to convert 96 column format, 3-1  
    number of columns on form, 1-8  
    order of entry, 1-7  
    types needed for sorts, 1-8  
    using \$STDIN, 1-6, 7-2  
    using XSORTTEXT, 1-6, 7-2  
SORT/3000, function with XSORT, 1-6  
SORT/3000, relationship to XSORT, 1-2  
SORT3  
    commands for converting to XSORT, A-1  
    compatibility with XSORT, A-1  
    correlation of formal names with XSORT names, A-2  
    unsupported program, 1-9  
SORTA (see Address-out)  
SORTC (see Count-only)  
SORTR (see Record-out)  
Spec type (record type specs, column 6), 5-3  
Special Option (S Option) specifications  
    discussion, 4-1  
    overview of column entries, 4-2  
    column entries, 4-2  
    column 1 to 5, line number, 4-2  
    column 6, S entry, 4-2  
    column 7, Address-out file first record number, 4-2  
    column 8, inplace sort override, 4-3  
    columns 10 to 16, maximum number of records to  
        be sorted, 4-3  
    when required, 1-8  
    with KSAM files, 1-10  
Syntax error messages, B-2

## T

Timing, factors affecting time required to run sort, 1-11  
Timing, comparison of address-out and record-out sorts, 1-5  
To entry, columns 13-16, field description specifications,  
    indicates where the field ends, 6-10  
To entry, columns 13-16, record type specifications,  
    indicates where factor 1 field ends, 5-10

To entry, columns 24-27, record type specifications,  
    indicates where factor 2 field ends, 5-11  
Type of comparison (record type specs,  
    columns 17-18), 5-10  
Types, record, 5-3

## U

Unconditional force, 6-8  
Unnumbered error messages, B-4  
Unpacked (U) data, definition, 5-8  
Unpacked constants, 5-12  
Unpacked numeric constants, 5-12  
Unpacked numeric data, maximum field length, 5-8  
Using XSORT commands, 7-1

## W

Work records, how they are built and formatted, 1-2  
Work records, type of data to be used (C/P/U/V), 6-9  
Work space, size determination with single-file input, 1-10  
Work space, when record numbers must be specified, 1-10

## X

XSORT alternate collating sequence coding and  
    worksheet, C-9  
XSORT, characteristics, 1-1  
XSORT compared to IBM System/3 \$DSORT and IBM  
    System 32 & 34 #GSORT, A-3  
XSORT, formal name, 1-9  
XSORTTEXT, definition, 1-6, 7-2  
XSORTOUT, output file formal file designator, 1-11





# PREFACE TO RPG INTERACTIVE SYSTEM ENVIRONMENT/3000

This publication is the reference manual for Hewlett-Packard's RPG Interactive System Environment/3000, also called RISE/3000.

RISE/3000 is a specialized editor designed for the creation and modification of RPG programs.

RPG is a commercial language whose usefulness has been established for years. Unfortunately, the necessary reliance on specification coding forms which include 80 columns and a highly specialized syntax can be time-consuming and tedious. RISE/3000 accelerates the process of creating source programs while maintaining necessary accuracy. It does so in a visually-oriented environment in which friendly interactive use is a fundamental feature.

In your use of RISE/3000 you may wish to consult the following Hewlett-Packard manuals:

Manual	Part Number
RPG Reference Manual	32104-90001
KSAM/3000 Reference Manual	30000-90079
MPE Commands Reference Manual	30000-90009
VPLUS/3000 Reference Manual	32209-90001

# CONTENTS

	page
<b>Section I</b>	
<b>INTRODUCTION</b> .....	1-1
<b>RISE Features</b> .....	1-1
<b>Requirements for Operation</b> .....	1-3
Hardware Requirements .....	1-3
Software Requirements .....	1-3
<b>User Interface with RISE/3000</b> .....	1-4
<b>Modes of Operation</b> .....	1-5
Line and Block Modes .....	1-5
Line Mode Advantages .....	1-6
Block Mode Advantages .....	1-6
Commands Restricted to Modes .....	1-6
Summary of Modes .....	1-7
<b>Features of Block Mode</b> .....	1-8
Command Menu .....	1-8
Command Window .....	1-9
Show Mode .....	1-9
<b>Special Function Keys (Softkeys)</b> .....	1-11
<b>Files</b> .....	1-12
The Work File .....	1-12
Accessing Files .....	1-13
Safeguards .....	1-14
<b>Running RISE</b> .....	1-15
<b>RISETOUR—a self-paced guided tour of RISE</b> .....	1-15
<b>RISE Commands</b> .....	1-17
Commands Divided by Mode .....	1-18
Commands Divided by Function .....	1-20
Command Limitations .....	1-21
<b>Section II</b>	
<b>COMMANDS</b> .....	2-1
<b>Notations for this Manual</b> .....	2-1
<b>Literal Parameters</b> .....	2-1
<b>Specifying the Sequence Numbers</b> .....	2-3
<b>Increments for Sequence Numbers</b> .....	2-4
<b>About the Examples</b> .....	2-5
<b>Commands—Detailed Descriptions</b> .....	2-8
<b>ADD Command</b> .....	2-8
Form .....	2-8
Parameter explanation .....	2-8
Purpose .....	2-8
Related commands .....	2-9
Leaving ADD .....	2-9
Examples .....	2-9

# CONTENTS (continued)

<b>BEGIN Command</b> .....	2-14
Form .....	2-14
Parameter explanation .....	2-14
Purpose .....	2-14
Examples .....	2-15
<b>CHANGE Command</b> .....	2-19
Form .....	2-19
Parameter explanation .....	2-19
Purpose .....	2-19
Special considerations .....	2-19
Delimiters .....	2-19
Examples .....	2-20
<b>COMMENT Command</b> .....	2-21
Form .....	2-21
Parameter explanation .....	2-21
Purpose .....	2-21
Special considerations .....	2-21
Examples .....	2-22
<b>COPY Command</b> .....	2-24
Form .....	2-24
Parameter explanation .....	2-24
Purpose .....	2-24
Examples .....	2-25
<b>DELETE Command</b> .....	2-26
Form .....	2-26
Parameter explanation .....	2-26
Purpose .....	2-26
Related commands .....	2-26
Examples .....	2-26
<b>EXIT Command</b> .....	2-28
Form .....	2-28
Parameter explanation .....	2-28
Purpose .....	2-28
Examples .....	2-29
<b>FILE Command</b> .....	2-30
Form .....	2-30
Parameter explanation .....	2-30
Purpose .....	2-30
Case 1: File must be created .....	2-31
Case 2: File exists but is not KSAM .....	2-31
Case 3: File exists and is KSAM .....	2-32
Related commands .....	2-32
Leaving FILE .....	2-33
Examples .....	2-33
<b>FIND Command</b> .....	2-39
Form .....	2-39
Parameter explanation .....	2-39
Purpose .....	2-39
Examples .....	2-40

# CONTENTS (continued)

<b>FORM Command</b> .....	2-42
Form .....	2-42
Parameter explanation .....	2-42
Purpose .....	2-42
Related commands .....	2-42
Examples .....	2-42
<b>GET Command</b> .....	2-45
Form .....	2-45
Parameter explanation .....	2-45
Purpose .....	2-45
Examples .....	2-47
<b>HELP Command</b> .....	2-49
Form .....	2-49
Parameter explanation .....	2-49
Purpose .....	2-49
Related commands .....	2-49
Examples .....	2-50
<b>INCREMENT Command</b> .....	2-52
Form .....	2-52
Parameter explanation .....	2-52
Purpose .....	2-52
Examples .....	2-52
<b>JOIN Command</b> .....	2-55
Form .....	2-55
Parameter explanation .....	2-55
Purpose .....	2-55
Examples .....	2-55
<b>KEEP Command</b> .....	2-58
Form .....	2-58
Parameter explanation .....	2-58
Purpose .....	2-58
Related commands .....	2-59
Examples .....	2-59
<b>LINE Command</b> .....	2-62
Form .....	2-62
Purpose .....	2-62
Related commands .....	2-62
Examples .....	2-62
<b>LIST Command</b> .....	2-63
Form .....	2-63
Parameter explanation .....	2-63
Purpose .....	2-63
Related commands .....	2-63
Examples .....	2-64

# CONTENTS (continued)

<b>MENU Command</b> .....	2-67
Form .....	2-67
Purpose .....	2-67
Related commands .....	2-67
Examples .....	2-68
<b>MODIFY Command</b> .....	2-70
Form .....	2-70
Parameter explanation .....	2-70
Purpose .....	2-70
Related commands .....	2-70
Examples .....	2-72
<b>MOVE Command</b> .....	2-76
Form .....	2-76
Parameter explanation .....	2-76
Purpose .....	2-76
Related commands .....	2-76
Examples .....	2-77
<b>PRINT Command</b> .....	2-79
Form .....	2-79
Parameter explanation .....	2-79
Purpose .....	2-79
Related commands .....	2-79
Examples .....	2-80
<b>RENUMBER Command</b> .....	2-81
Form .....	2-81
Parameter explanation .....	2-81
Purpose .....	2-82
Examples .....	2-82
<b>RUN Command</b> .....	2-86
Form .....	2-86
Parameter explanation .....	2-86
Purpose .....	2-88
Examples .....	2-88
<b>SHOW Command</b> .....	2-91
Form .....	2-91
Parameter explanation .....	2-91
Purpose .....	2-91
Related commands .....	2-92
Examples .....	2-95
<b>TEXT Command</b> .....	2-104
Form .....	2-104
Parameter explanation .....	2-104
Purpose .....	2-104
Related commands .....	2-105
Examples .....	2-105
<b>UNDELETE Command</b> .....	2-108
Form .....	2-108
Purpose .....	2-108
Related commands .....	2-108
Examples .....	2-109

# CONTENTS (continued)

<b>VERIFY Command</b> .....	2-111
Form .....	2-111
Parameter explanation .....	2-111
Purpose .....	2-113
Related Commands .....	2-114
Use of Split Screen .....	2-114
Examples .....	2-115
<b>XPAND Command</b> .....	2-122
Form .....	2-122
Parameter explanation .....	2-122
Purpose .....	2-122
Related commands .....	2-122
Examples .....	2-123
<b>MPE Commands</b> .....	2-124
Form .....	2-124
Purpose .....	2-124
Related commands .....	2-124
Examples .....	2-124
<b>Section III</b>	
<b>RECOVERY PROCEDURES</b> .....	3-1
Recovery from System Failure .....	3-1
Recovery from Terminal Lockout .....	3-2
Recovery from Breaking out of Block Mode .....	3-3
Recovery from Transmission Error .....	3-3
Recovering echo after it is lost .....	3-4
Recovery from :EOD (End of Data) .....	3-4
<b>Appendices</b>	
<b>APPENDIX A</b> .....	A-1
Command Summary .....	A-1
<b>APPENDIX B</b> .....	B-1
Comparison of RISE and Edit/3000 .....	B-1
<b>APPENDIX C</b> .....	C-1
Informative Execution Messages .....	C-1
<b>APPENDIX D</b> .....	D-1
Error Messages .....	D-1

# ILLUSTRATIONS

Rise/3000 Environment .....	1-2
Internal/External Management .....	1-3
HP 2645 Terminal Keyboard .....	1-5
RISE/3000 Editing Methods .....	1-13





# INTRODUCTION

SECTION

I

The RPG Interactive System Environment/3000 is a specialized editor, designed to create and modify programs written in the column-dependent language, RPG.

It is an RPG Utility and, therefore, intended for the use of RPG programmers. You may need it for a simple, quick program, or you may have a job which requires interfacing with KSAM, Image, and VPLUS/3000. However extensive your experience, however technical the job at hand, this system environment will prove a useful, productive, and convenient tool for your programming needs. Throughout this manual, the RPG Interactive System Environment/3000 will be referred to as RISE for the sake of simplicity.

## **RISE offers these features:**

- visual editing with images that represent the RPG specification coding forms.
- menu and special function keys (softkeys) for a broad range of uses.
- page-at-a-time direct screen editing via the terminal's cursor control keys.
- ability to call the RPG/3000 compiler as well as the Segmenter and manipulate the compiled listing using a split screen.
- choice of editing a file directly or editing a copy of that file.
- ability to execute important MPE commands and run any program file.

- HELP facility which presents explanations of individual commands or a brief description of all commands.
- ability to create a comment banner which is presented visually apart from the rest of the source program in which you type your comments.
- renumbering command which also allows you to renumber the lines (columns 1-5) of the RPG source program.
- ability to execute commands stored in a file.
- ADD mode which allows you to change RPG specification forms, and to DELETE, LIST, or MODIFY lines while in ADD mode.
- recoverability features including an UNDElete command and permanent work files which are protected from system failures.

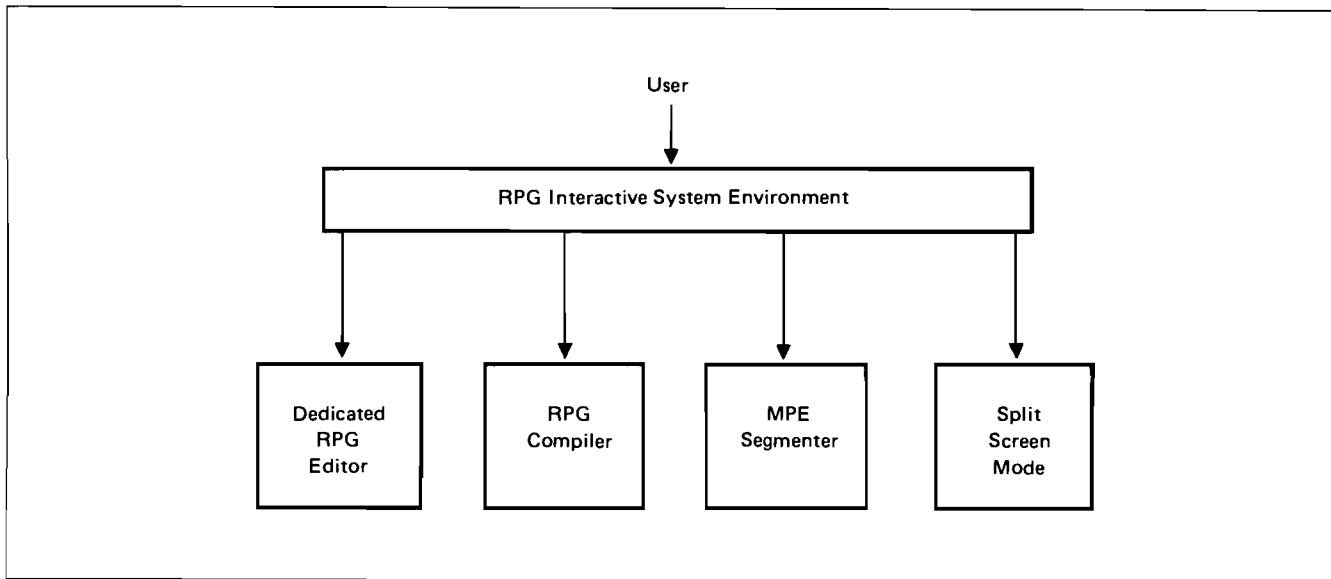


Figure 1-1. RISE/3000 environment.

# REQUIREMENTS FOR OPERATION

## *Hardware Requirements*

RISE operates as a member of the VPLUS/3000 family. Your terminal must be one of those compatible with VPLUS/3000, such as the model 2645.

In addition, your terminal must have at least 8K of memory and the display enhancement character set. VPLUS/3000 requires a minimum of 8K of memory in your terminal to store and display RISE's special forms of the RPG Specification Records. These forms utilize the display enhancement character set to the user's advantage.

## *Software Requirements*

Software required to run RISE consists of MPE IV, V/3000, and KSAM/3000. RISE uses VPLUS/3000 for its user interface and KSAM/3000 for internal management of the file you are editing.

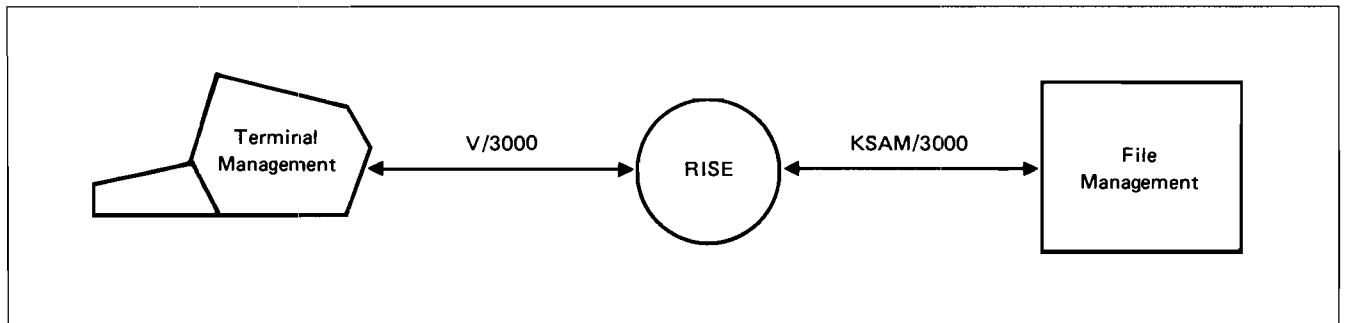


Figure 1-2. Internal/External management.

# USER INTERFACE WITH RISE/3000

Friendly interface was a guiding principle in the design of RISE. The stress on visual integration of the RPG specification forms makes it much simpler to enter RPG specifications into the system.

Initially, RPG programs are written out on specification coding forms. This allows the user to review the semantics of each column and ensure that they contain the entries necessary to accomplish their data processing tasks and produce the output desired. Entering specifications without a form display can be tedious and confusing, and may require extensive column counting.

RISE allows you to place the appropriate form on the screen for the entry of associated specifications. Protected areas of the screen — those which cannot be changed or modified by the user — contain information and graphic guides which make entering more convenient.

The RPG specification forms available for display are Header Specifications, File Specifications, File Extension Specifications, Line Counter Specifications, Input Specifications, Calculation Specifications, and Output Specifications. You can also use a column numbering display known as a column indicator. Using these forms you can edit, add to, or remove specifications directly on the displayed form.



## MODES OF OPERATION

### *Line and Block Modes*

RISE operates in Line or Block mode using the terminal's powerful capabilities to their fullest. Each mode offers you its own unique advantages. Line mode is similar to Edit/3000. You type in lines in response to a ">" prompt. Typing errors can be corrected by backspacing, which erases the errors and allows you to re-type the line as it should appear. When you press the carriage return you send the line to RISE for execution.

Block Mode resembles VPLUS/3000. You enter anything you wish on the screen in unprotected data fields. You may correct typing errors or alter the information displayed as needed. You position the cursor, using the cursor control keys and editing keys to make your changes directly on the screen. Afterwards, you press the ENTER key to send all the data on the screen to RISE for processing.

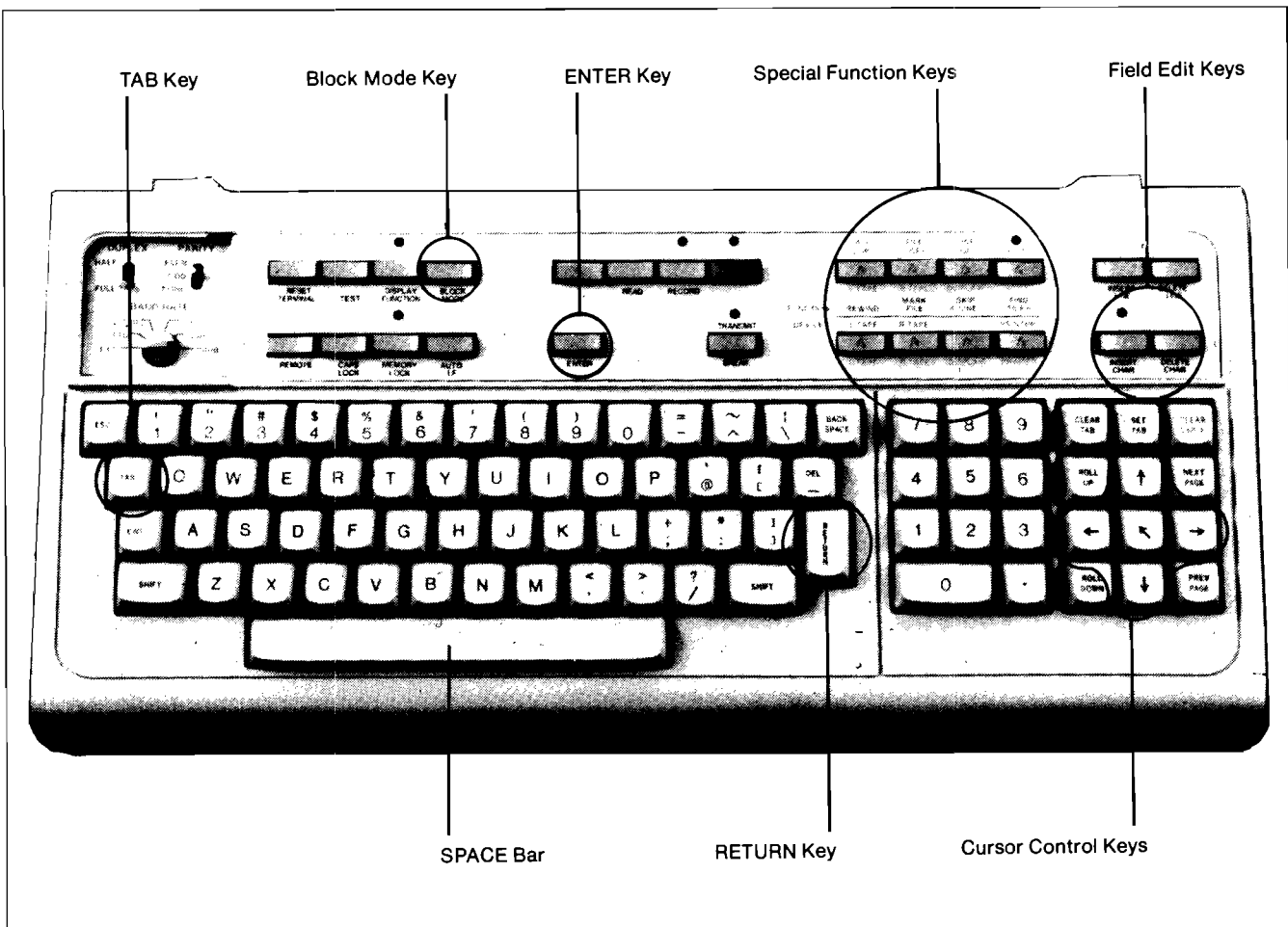


Figure 1-3. HP 2645 Terminal Keyboard

### ***Line Mode Advantages***

RISE is primarily a line mode editor. Line mode is effective because of the speed with which commands, one at a time, are sent to RISE for execution. Also, the effects of a command's execution are shown to you on the screen so that you see exactly what has occurred. For example, when you give the DELETE command and press the carriage return, RISE will display the lines being deleted.

Another advantage of Line Mode is its use of the terminal's display memory. Using the LIST command you can fill the screen and memory with text, then scroll up or down, or call the next or previous page for review, by pressing the terminal's special keys.

### ***Block Mode Advantages***

In Block Mode, you can make numerous changes to the data on the display and send them to RISE/3000 by pressing the ENTER key once. Your special function keys (softkeys) are activated to give you what is, effectively, pushbutton control. The purpose of the special function keys varies with Block Mode commands you have entered. Descriptive softkey labels appear at the top of the screen with each change of key function.

### ***Commands Restricted to Modes***

Certain commands operate only in Line Mode; others are restricted to Block Mode. Some work in either mode. If you give a command pertaining to one mode while in the other, RISE will automatically shift to the correct mode for that command. For example, if you enter the SHOW command while in Line Mode, RISE immediately shifts to Block Mode, and a display of RPG source lines will appear on your screen, allowing you to make any changes directly. Thus, all commands may be entered while in either mode.

## Summary of Modes

	<b>Line Mode</b>	<b>Block Mode</b>
reached by	Special Function Key and automatic shift when you enter line command	automatic shift when you enter Block Mode command
commands entered by	Carriage Return	ENTER Key
advantages	line-by-line editing	full screen display with RPG forms
	commands executed with greater speed	full screen editing with single entering procedure
	rapid, convenient editing methods	Special Function Keys for push-button control

Since RISE's command set is similar to that of EDIT/3000, familiarity with the editor will be of value in using RISE. It is not necessary, however. (See Appendix A for a comparison of RISE and Edit/3000.)

# FEATURES OF BLOCK MODE

## Command Menu

RISE offers you a command menu which lists the most important RISE commands so that you can select the next command you need. The commands are grouped by their functions: File Commands, Editing Commands, General-1 Commands, and General-2 Commands. You move the cursor to the group using a special function key labeled with the group title. Windows are provided for the parameters of each command.

The menu relieves you of the need to memorize or look up the commands since it displays the options for a command. The menu operates only in Block Mode.

```
(1) FILE CMDS   GEN1 CMDS   SEQNUM * OFF *   COMMAND WINDOW   EDIT CMDS   GEN2 CMDS   LINE MODE   REFRESH MENU

F1: FILE COMMANDS                               F8: GENERAL1 COMMANDS
█ - █ - █ - █ - █                               █ - █ - █ - █ - █
File-filename ->B..                               Incr - value
Text-filename ->L..                               Print-from ln - to line - PAGELIST
Keep-filename ->C..                               Com ->C - at line - by inc
Join-filename ->A line ->B by and                 Renum-from ln - to line - by inc
Move -from ln - to line - at line - by inc

F2: EDITING COMMANDS                             F9: GENERAL2 COMMANDS
█ - █ - █ - █ - █                               █ - █
Delete- from line- to line
Show - from line- to line
Add - at line - by inc
Find - string - from line- to line
Copy - from line- to line - at line - by inc
Change- oldstring- newstring- from ln - to line

COMMAND: [ ]
ENTER
COMMAND MODE
```



## ***Command Window***

Besides the Command Menu, you have another means of entering commands while in Block Mode. You can use the Command Window which is always displayed on the terminal when you are in this mode.

The Command Window is useful to those who are familiar with the commands and have no need to go through the menu but want immediate action. In such an instance, you enter the command in the window as you would enter it in Line Mode, using a simple syntax which is described in Section II.

A Block Mode enter prompt (ENTER?) appears to the bottom left of the Command Window. This tells you that RISE is waiting for your next command. The enter prompt disappears when a command is being executed and returns upon completion.

## ***Show Mode***

The SHOW command will display an RPG Record Specification form at the top of the screen, followed by a page (10 lines) of RPG source records in Block Mode. If you have changes you wish to make to the records, move the cursor to those places where the changes are to be made. Pre-set tabs allow you to use the Tab key to skip quickly across the source records to important columns.

When editing, you directly enter your changes on the screen so that what is visible duplicates exactly what will be stored in the file. When your editing is finished, you press the ENTER key to send your updated page to RISE. If, at that time, you want to continue with the next page of the source file, you push the special function key, F5 (Scroll forward). If you wanted to go into Line Mode, you would push F7.

The Command Window can be used with displayed pages in Show Mode, enabling you to perform several editorial functions at the same time. For example, you may wish to modify the first page of text while deleting lines 100/150. To do this you would do the following:

- SHOW the first page (10 lines) of the source code for the modification. (Your Command Window will also appear on the screen.)
- Type in your modifications to the program directly.
- Type in DELETE 100/150 in the Command Window.
- Press the ENTER key.

After you press ENTER, RISE reads in your changes, then parses and executes your DELETE command, leaving you in the page modification mode of the SHOW command.



## SPECIAL FUNCTION KEYS (SOFTKEYS)

The terminal special keys are fully integrated into the functional features of RISE. The editing keys and cursor positioning keys allow you to edit directly a displayed page. The special function keys give you push-button control of scrolling, mode shift, and a display of editor sequence numbers, among other capabilities. The function of the keys changes, depending upon the command which has been entered.

You do not have to memorize the function of the softkeys; descriptive labels associated with the activated keys appear at the top of the screen whenever the keys are available for use.

In the commands section of the manual (See Section II), the special function keys, also called softkeys, are discussed with the commands that bring them to the screen.

# FILES

## *The Work File*

RISE performs all of your editing commands on a work file. The File and Text commands tell RISE which permanent file you wish to become your work file. The work file is a KSAM file (Keyed Sequential Access Method), and is used so that RISE can quickly random access your source records. (See the KSAM/3000 Reference Manual for more information on this file organization.)

The name of the work file is in the form Wdddhhmm, where "ddd" is the day of the year, "hh" is the hour of the day, and "mm" is the minute of the hour. The name of the associated key file is WKdddhhmm, where "K" overlays the first "d" in the day of the year, and "ddhhmm" have the same meaning as they do with the work file. If there is an existing file with the same name, RISE will increment the last digit.

RISE automatically supplies sequence numbers to the KSAM work file in columns 81-88 if none already exist. These sequence numbers serve as search keys when RISE accesses the KSAM work file. In your command parameters you specify which source lines you want to work on. For example "LIST 1/10" will list to the screen the source lines whose sequence numbers are included in the range 1 to 10.

The RENUM [ber] command allows you to manage the sequence numbers. Note that the sequence numbers are not associated with the RPG line numbers in columns 1-5 of your source program. The sequence numbers are not a part of the RPG program.

There are several ways in which you may specify which sequence numbers you want processed. You may give the precise number, as in "LIST 1/10", or you may use relative numbers in relation to the current position of the record pointer. You may also use the literal specifications FIRST, LAST, or ALL. More information concerning the specification of sequence numbers with various commands appears in the command section (Section II).

## Accessing Files

When you TEXT your files, your permanent file is copied into a work file (which is a KSAM file). It is copied back to a permanent file when you use the KEEP command. Using TEXT and KEEP you edit work files only, bringing them to the screen for editing and storing them as permanent files when the work is concluded.

The TEXT/KEEP commands provide a safeguard in that the editing done to a work file is not made permanent until the KEEP is given. Thus, if erroneous deletions were made to the work file, you would retain the deleted lines in the permanent file. Using the TEXT command you could copy it back to the work file in its original state for further work at the terminal.

The price of the safeguard is additional time required to copy the files back and forth between your permanent file and the RISE work file. If your RPG source file is long, the overhead may be considerable.

You can also edit your files directly using the FILE/EXIT commands. Only one copy of the file is created with this option. RISE edits the copy directly so that no time is wasted.

In using FILE/EXIT you sacrifice your safeguard, however. You do not have a permanent copy which can be recalled if, for example, you wish to restore lines mistakenly deleted. Changes made to a file called with the FILE command cannot be rescinded unless you made a backup copy before editing it.

In general, FILE/EXIT serve best when you are adding to or editing a large RPG source file. When your file is small, TEXT/KEEP are recommended. Also, for a more efficient use of space, TEXT/KEEP are recommended for large files that are debugged, complete, and in production. Such files rarely need additional editing. Storage of a regular sequential file takes less disc space than storage of the same file in KSAM format because of the extra space consumed storing the associated key file.

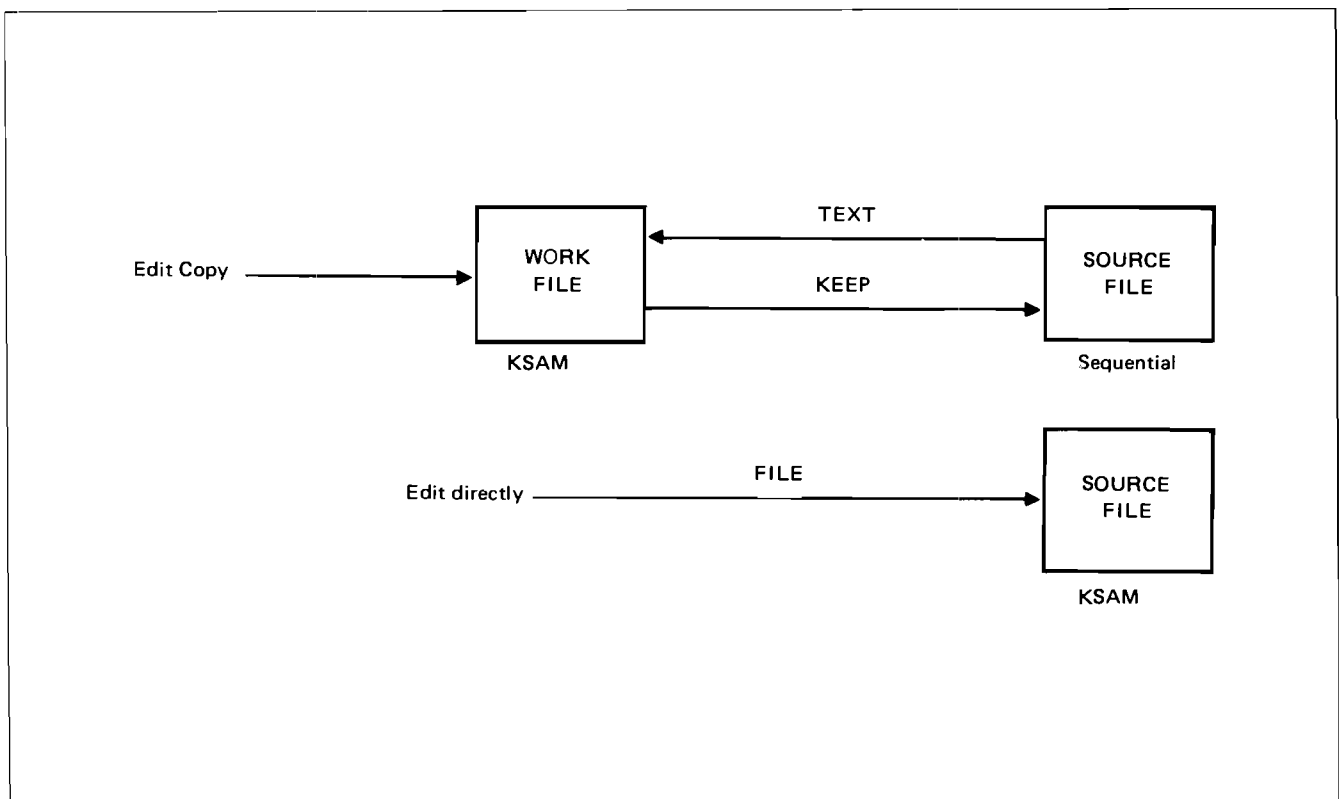


Figure 1-4. RISE/3000 editing methods.

## SAFEGUARDS

RISE has two important built-in safeguards which help to protect you from two serious errors that you may accidentally make. The first safeguard is associated with the TEXT, FILE, EXIT, and BEGIN commands. These commands always clear the work file, and there is a danger that you might clear or destroy a work file you have been editing because you entered one of these commands without first keeping your recent changes to a permanent file. When there is a possibility that this might happen, RISE will prompt you with the following question:

“KEEP not done. CLEAR current work file?”

Thus you are reminded that the changes you have made while editing will not be saved if your TEXT, FILE, EXIT, or BEGIN is immediately executed. You are given the opportunity to decide whether or not you wish to save the changes first. You may respond “NO” to the question if you want to cancel the command that you entered; or you may respond “YES” if you do want RISE to go ahead and clear the work file, executing your command without saving your changes.

The second safeguard is associated with the KEEP command. It protects you from overwriting an existing permanent file. If you try to KEEP the contents of your work file to a name of a file that already exists, RISE will prompt you as follows:

“File exists already. Purge old 'filename'?”

This gives you the opportunity to decide whether or not you want to overwrite the pre-existing permanent file. If you do, you respond “YES” and RISE will execute your KEEP command. If you do not want to lose the contents of the pre-existing file, you respond “NO” and RISE will cancel the execution so that you can KEEP to a different file name if you wish.

There will be occasions when you have decided beforehand that immediate execution of the TEXT, FILE, EXIT, BEGIN, and KEEP commands is precisely what you want. In such an instance, the prompts would cause unnecessary delays. You have an optional parameter called “NOW” which saves time by disabling RISE’s safeguards. If you specify “NOW” with any of these five commands, RISE will execute immediately without prompting you to reconsider. You use the parameter when you are certain of the results you want.

## Running RISE

To run RISE you enter the following MPE command at your terminal:

```
RUN RISE.PUB.SYS
```

Your command will be answered with an HP banner like the one below:

```
HP32104A.05.00  RPG INTERACTIVE SYSTEM ENVIRONMENT  RISE/3000  
(C) HEWLETT-PACKARD CO. 1981          WED, SEP 16, 1981, 10:29 AM  
>
```

With that you have begun the program and entered the RPG Interactive System Environment. Other MPE commands are available for your use within the system. A list of these commands may be obtained by typing "HELP:". The commands are displayed on the screen in Block Mode.

To leave the system you use the EXIT command.

Because of the visual and interactive emphasis in the design of RISE, it is not recommended that you use it in batch mode.

## RISETOUR — A Self-paced Guided Tour of RISE

RISETOUR is a self-paced introduction to RISE. It is on-line and can be called by the following commands:

```
:RUN RISE.PUB.SYS
```

```
HP32104A.05.00  RPG INTERACTIVE SYSTEM ENVIRONMENT  RISE/3000  
(C) HEWLETT-PACKARD CO. 1981          WED, SEP 16, 1981, 10:29 AM  
>
```

```
>GET RISETOUR.PUB.SYS
```

However, before you initiate RISE TOUR, you must have the file SIMCAL in your group and account. SIMCAL is an RPG source program referenced in RISE TOUR (and used extensively in the examples included in this manual). To get a copy of SIMCAL into your group and account, you can use FCOPY to copy it from "SIMCAL.PUB.SYS" to "SIMCAL.yourgrp.youracct".

RISE TOUR is interactive; it contains many requests and directions which call for your direct participation in the tour. It is automatic and controls the complete tour. Using it, you will quickly develop a familiarity with most of RISE's features first-hand.

The tour is divided into lessons, and the lessons are grouped under the following headings:

- File manipulation commands
- General editing commands
- Block mode
- RPG dedicated commands
- Calling the RPG/3000 compiler

RISE TOUR is designed as an introduction rather than an intensive training course. Messages are provided to explain the relationship between what appears on the screen and its application to the needs of an RPG programmer. Pauses are imbedded to allow time for you to study the examples on the screen.

RISE TOUR can also be useful as a printed reference guide. If you text in "RISE TOUR.PUB.SYS" and use the "LIST ALL OFF" command, you will get a listing of the tour offline on a line printer. This will allow you to review at length parts of the tour without the delays caused by scrolling through the screen version.



## RISE Commands

RISE only accepts commands that are typed in capital letters. This restriction will prevent you from accidentally entering your RPG source code in lower case letters, which the RPG/3000 Compiler will not accept.

These commands are listed below with brief explanations of their use. The portions of the commands outside the brackets are the minimum abbreviations that RISE will recognize and accept.

Futhermore, any number of letters in the optional portion of a command enclosed within brackets may be given and will be recognized by RISE. For example, in the "E [XIT] " command, the "E" is required and the "XIT" portion is optional. The following variations are all acceptable:

E  
EX  
EXI  
EXIT

## Commands List

1. A [DD] —add new lines
2. B [EGIN] —start editing a new work file
3. CH [ANGE] —change oldstring to newstring
4. COM [MENT] —create a comment banner for user documentation
5. COP [Y] —duplicate lines
6. D [ELETE] —delete lines
7. E [XIT] —end subsystem
8. FIL [E] —work file directly
9. F [IND] —locate a string
10. FO [RM] —display an RPG specification form on the screen
11. G [ET] —read and execute commands from a file
12. H [ELP] —explain commands
13. I [NCR] —set default increment value
14. J [OIN] —append or merge an entire file
15. K [EEP] —save the work file
16. LIN [E] —enter Line Mode
17. L [IST] —list lines in Line Mode
18. ME [NU] —display main menu
19. M [ODIFY] —modify lines
20. MOV [E] —move lines to new location
21. P [RINT] —print lines offline
22. R [ENUM] —renumber editor sequence numbers or RPG source line numbers
23. RU [N] —run a program file
24. S [HOW] —display a page in Block Mode for direct screen editing
25. T [EXT] —copy a file into a work file for editing
26. U [NDEL] —restore lines deleted by last DELETE command
27. V [ERIFY] —compile the work file or LIST the compilation listing or prepare a USL file
28. X [PAND] —expand the work file size
29. :MPEcommand —execute MPE command

The RISE commands can be grouped together by the mode in which they execute and the functions they fulfill. It is not necessary to memorize the modes in which commands operate. If you give a command unsuited to the mode in which you are operating, RISE will automatically shift you to the mode that will accept and execute it. The HELP command provides additional convenience by listing and explaining RISE commands on the screen.

The following commands execute only in Line Mode:

ADD	BEGIN	FORM	LINE
LIST	MODIFY	RUN	:MPEcommand

The following commands execute only in Block Mode:

COMMENT	HELP	MENU	SHOW	VERIFY
---------	------	------	------	--------

The following commands execute in either mode:

CHANGE	COPY	DELETE	EXIT
FILE	FIND	GET	INCR
JOIN	KEEP	MOVE	PRINT
RENUM	TEXT	UNDEL	XPAND

The following commands, if entered in Block Mode, will automatically return you to the main menu when the command is completed:

FILE	RENUM	TEXT
------	-------	------

These commands alter the internal state of the work file. What shows on the screen after they have been entered in Block Mode is not representative of what has occurred internally. The main menu resets the screen contents so that they are representative.

For example, suppose you are in Show mode with the first page of File1 displayed on the screen and find it necessary to edit File2. You would then enter the command "TEXT FILE2". After RISE completes the execution, your new internal work file will contain the contents of File2. However, the screen will still be displaying the first page of File1. At this point, RISE will automatically clear the screen by displaying the main menu. Then you may enter the Show command to have the first page of File2 displayed.

## Commands Divided by Function

The following commands are grouped by function. The division presented here appears on the main menu. See MENU for a description of how you use the cursor and softkeys to execute the commands from the main menu. Note that not all commands appear on the main menu, but all the more important ones do.

### F1: FILE COMMANDS

- FILE
- TEXT
- KEEP
- JOIN

### F2: GENERAL1 COMMANDS

- INCRement
- PRINT
- COMment
- RENUMber
- MOVE

### F5: EDITING COMMANDS

- DELETE
- SHOW
- ADD
- FIND
- COPY
- CHANGE

### F6: GENERAL2 COMMANDS

- UNDELete
- HELP
- VERIFY
- EXIT

## Command Limitations

RISE will not accept commands longer than 50 characters. This limitation is established by the number of characters which will fit into the Command Window. It holds true with all commands, whatever the mode in which they are entered.

Also, multiple commands entered on a single line are not accepted by RISE.

RISE does not accept special characters used to separate parameters for a command. This is a convenience designed to relieve you of the need to remember which special characters to use for delimiters. For example, "LIST ALL OFF" is correct; "LIST ALL, OFF" is not because of the special character (the comma).

It is necessary, however, to use the space bar to separate the parameters with one or more blank spaces except when the parameters are of a different type (alphabetic vs. numeric). For instance, in the command "FIND 'string' IN FIRST", the space separating IN and FIRST is required since both parameters are alphabetic. If there was no space, RISE would not be able to recognize the parameters. However, the command "FIND 'string' IN 1", would be accepted without a space separating IN and 1 because the parameters are of different types and RISE readily distinguishes between the two.

For your convenience, RISE will accept the F [ORM] and COM [MENT] commands without a space between the command and the alphabetic parameter. Whenever RISE encounters an unacceptable use of parameters, an error message appears on the screen to explain the problem.



## Notations for this Manual

Notation	Meaning	Example of Use
[opt]	part of command or parameter enclosed by brackets is optional	A [DD] [^^ [formf] ]
form	one of the designators of the RPG Record Specifications—H,F E,L,I,C,O—or CO [L] , which brings a column indicator to the screen	FO [RM] form
lb	line begin (beginning with line)	D [ELETE] lb
le	line end (ending with line)	D [ELETE] lb/le
inc	increment value	A [DD] BY inc
/	delimiter with semantics of TO	L [IST] lb/le
X	choose one of the options, X or Y	A [LL]
Y		L [IST] lb/le
<b>Literal Parameters</b>		
BY	the following number is an increment value	A [DD] BY .01
CO [L]	display with column indicator	FO [RM] CO [L]
TO	the following line number is a destination;or, the following is a new string	COP [Y] I TO I0 CH [ANGE] "OLD" TO "NEW"
IN	the following is a range of line numbers	F [IND] "string" IN lb
E [VERY]	locates every occurrence of string in the FIND command	F [IND] E [VERY] "string" IN A [LL]

O [FF]	list lines offline on a line printer in the LIST command	L [IST] A [LL] O [FF]
N [OW]	the command is to be executed immediately without safeguards	E [XIT] N [OW]
U [NN]	the work file is to be made permanent without the appended sequence numbers	K [EEP] SAVEFILE U [NN]
A [LL]	the command applies to the entire work file	D [ELETE] A [LL]
F [IRST]	indicates first line in work file	COM [MENT] F [IRST]
L [AST]	indicates last line in work file	M [ODIFY] L [AST]
^^	controls the display of the RPG Record Specification form	A [DD] ^^H
R [EPEAT]	controls the auto-repeating factor in ADD mode	A [DD] ^^H R [EPEAT] 3
*	indicates current position of record pointer (in Line Mode only)	L [IST] *-5/*+5
@	may be used in place of *	L [IST] @-5/@+5

Those command descriptions which include references to <Control Y> and the use of "\*" for the record pointer position apply only if RISE is in Line Mode.



## Specifying the Sequence Numbers

There are two ways of specifying sequence numbers: you can use references to actual line numbers or you can refer to them relative to the current position of the record pointer. The record pointer is located at your current editing sequence number in the file. In this instance, only the pointer's position relative to sequence numbers is used.

Below is a summary of the ways in which you may specify sequence numbers using RISE:

Actual specification	<p>L20/37 (In the manual notation, this will be written as "lb/le", meaning line beginning and line ending.)</p> <p>F [IRST] /L [AST] may be used in place of lb/le</p> <p>A [LL] may be used in place of lb/le</p>
Relative specification (The special character, *, indicates the current position of the record pointer.)	L*-5/*+5
(The special character, @, may be used in place of * to indicate record pointer position.)	L@-5/@+5
(Specifications may be mixed.)	<p>L F [IRST] /* + 10</p> <p>L *-5/L</p> <p>L 10/L</p>

## Increments for Sequence Numbers

Every text record in RISE's work file has a unique sequence number that RISE uses to access the KSAM work file. The difference in value between each sequence number is known as the increment. RISE uses a default increment value of 1 which you may replace with the INCR [ment] command followed by the new value, so long as that value is between .001 and 5000 inclusively.

Those RISE commands which create new sequence numbers and text records for you need an increment value to determine the next new number. Some of the commands involved are RENUM, ADD, COPY, MOVE, and JOIN. Each of the commands provides the optional parameter "BY inc" to allow you to specify the increment for RISE to use while executing the command. If you omit "BY inc", RISE will use the current default increment value.

At times you will want to add or merge new text records between two existing sequence numbers. If the current increment value that RISE is using is too large, the new sequence numbers being generated may exceed the existing upper sequence number. If this happens, RISE will automatically make room for your new lines by reducing the large increment to a smaller one to stay within the upper limit while keeping your text records in proper order.

If, however, RISE cannot determine a small enough increment value because you are attempting to introduce too many lines into a restricted space between two existing line numbers, RISE will display the following error message:

Error 144: No room between line numbers.

In such a case, you can use the RENUM [ber] command to change the sequence numbers and continue with your editing.

## About the Examples

Examples of commands in the section which follows use a short RPG source file called SIMCAL (SIMple CALculator). The program will perform simple calculations—addition, subtraction, multiplication, and division. Here is a listing of the file which was made using RISE's PRINT command.

```

1      00011H                                                    SIMCAL
2      00012FINPUT      IP  F      72      DISC
3      00013FOUTPUT    O   F      72      DISC

4      00014IINPUT      AA  01      1 CA
5      00015I          OR   02      1 CS
6      00016I          OR   03      1 CM
7      00017I          OR   04      1 CD
8      00018I          OR   05
9      00019I
10     00020I                    3      72OPRND1
                                9      132OPRND2                    99

11     00021C  01      OPRND1  ADD  OPRND2  RESULT 104
12     00022C  02      OPRND1  SUB  OPRND2  RESULT
13     00023C  03      OPRND1  MULT OPRND2  RESULT
14     00024C  04N99  OPRND1  DIV  OPRND2  RESULT  H
15     00025C  04 99  OPRND1  Z-ADD0  RESULT

16     00026OOUTPUT    H 22      1P
17     00027O
18     00028O
19     00029O          D 11      N1P
20     00030O
21     00031O                    OPRND1
22     00032O                    01
23     00033O                    02
24     00034O                    03
25     00035O                    04
26     00036O                    OPRND2
27     00037O                    RESULT
28     00038O
29     00039O          D 1      1P
30     00040O          OR      N1P
31     00041O
32     00042O
33     00043O          D 11      1P
34     00044O          OR      N1P
35     00045O
36     00046O          D      05
37     00047O
38     00048O*END OF PROGRAM

11 "SIMPLE "
33 "CALCULATOR"

19 ". . . . . YOUR PROBLEM:"
27 " 0 ."
32 "ADD "
32 "SUB"
32 "MULT"
32 "DIV "
39 " 0 ."
46 "EQUALS"
60 " , 0 . -"

10 "ENTER DATA"
25 "IN THE FORMAT:"

13 "C NNNDD NNNDD"
15 "INVALID REQUEST"

```

To compile, prepare, and run SIMCAL from within the RPG Interactive System Environment, you would enter the commands below. The comments on the right are for your information and are not part of your responses. While the Segmenter is executing, its HP banner and responses will appear on your screen. You are expected to supply colon (":") prompts as required when entering the ":MPEcommand".

>T [EXT] SIMCAL	Copy SIMCAL to work file
>V [ERIFY] R [PG]	RPG compile the work file into a USL file
>PRINT ALL RPGLIST	Prints compilation listing offline
>V [ERIFY] P [REP]	Prepare USL to program file
>:FILE INPUT=\$STDIN	Issue file equations for I/O
>:FILE OUTPUT=\$STDLIST	
>RU [N] \$OLDPASS	Execute SIMCAL

Alternatively, you could prepare SIMCAL manually using the commands listed below. Again, you will have to supply colon prompts where they appear below. These are examples of the ":MPEcommand". (See :MPEcommand.)

RISE will respond with execution messages, mode changes, and banners to some of the commands. The example below includes only the prompt for information to be used in running the calculator. Comments appear to the right and are not part of your responses. Also, see the MPE Segmenter Reference Manual, part no. 30000-90011, for more information on USL and program files.

>T [EXT] SIMCAL	Copy SIMCAL to work file
>L [IST] A [LL]	List contents for review
>V [ERIFY] R [PG]	RPG compile the work file
>PRINT ALL RPGLIST	Prints compilation listing offline
>RU [N] SEGDVR.PUB.SYS	Run the Segmenter subsystem
-USL \$OLDPASS	Identify the USL file
-PREPARE \$NEWPASS	Prepare USL file in a new program file
-EXIT	Leave the Segmenter subsystem
>: FILE INPUT=\$STDIN	Issue FILE equations for I/O
>: FILE OUTPUT=\$STDLIST	
>RU [N] \$OLDPASS	Execute SIMCAL

When SIMCAL executes, you will receive the following message:

```
SIMPLE  CALCULATOR
```

```
Enter the data in the format:  
C  NNDD  NNDD
```

Under C, you enter S (subtract), M (multiply), D (divide), or A (add). Your operands may include three whole numbers (NNN) and two decimals (DD).

After you enter the data in the prescribed format, SIMCAL produces the arithmetic answer and prompts for more data.

When you have all the calculations you require, type ":" (colon) and you will exit from SIMCAL and return to RISE. Do not type ":EOD" to end SIMCAL for it will also end RISE.

# COMMANDS: DETAILED DESCRIPTIONS

## ADD

ADD allows you to add new lines to a work file.

### Form

```
A [DD] [lb] [BY inc] [^^ [form] [R [EPEAT] [n]]]
```

### Parameter explanation

- lb** The line number where you wish to start adding lines. If not specified, adding begins after the last line in the work file. If "lb" exists in the work file, RISE will first display the existing source record at "lb" and prompt you with a sequence number immediately following "lb".
- BY inc** Increment by which you wish to add lines. If omitted the current default is used.
- ^^** Your control of the RPG Record Specification forms. By itself "^^" homes the cursor and clears the screen before you begin adding lines. The "^^" option and its parameters may be entered while in ADD mode at the start of a new line to change the RPG Record Specification form and/or the REPEAT option. (See R [n] below.)
- form** Additionally displays the RPG Record Specification form type you designate. "form" may be H,F,E,L,I,C,O or CO [L].
- R n** The REPEAT command homes the cursor and clears the screen of all but the Record Specification form or column indicator after you have entered "n" number of lines. If "n" is omitted or "n" equals zero, the auto repeat is turned off.

Note: REPEAT parameter must be preceded by "^^".

### Purpose

The ADD command allows you to add new lines to a work file. If there is no work file established — if you have not entered a TEXT or FILE command — your ADD command will automatically create a new work file.

## Related commands

While in ADD mode, you may enter the LIST, MODIFY, and DELETE commands. To do so, you type ">>" followed by the command at the start of a new line.

If you type ">>" without a command, the cursor moves to column 6 of your RPG Record Specification form. This relieves you of the tedious task of spacing over columns 1 through 5, where the RPG optional line numbers are specified, in order to begin entering your record specifications.

Once you type ">>" alone at the start of a new line, RISE will always prompt you for new text starting from Column 6 whenever you are in ADD mode.

To return the cursor to column 1, type "<<", and RISE will prompt you from there.

## Leaving ADD

To leave the ADD mode you type <Control Y> or "/" at the start of a new line.

Execution mode: Line

Record pointer: At last added line

## EXAMPLES

The following are legal abbreviations used with the ADD command and its parameters:

A10 ^^H R3          Add line 10, Header Specifications form, REPEAT 3.

A ^^H                Add under Header Specification form.

AD L BY 10         Add beginning after last line using an increment value of 10.

Examples in this manual are based upon the file SIMCAL. In the examples of the ADD command, a portion of SIMCAL is entered. Various parameters are used. Errors are committed and the messages appear with the listing.

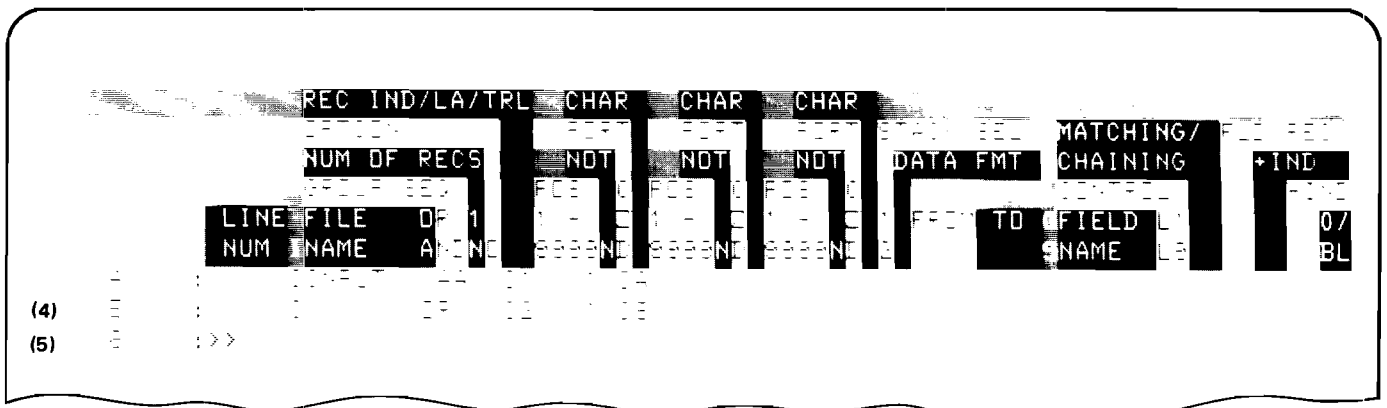
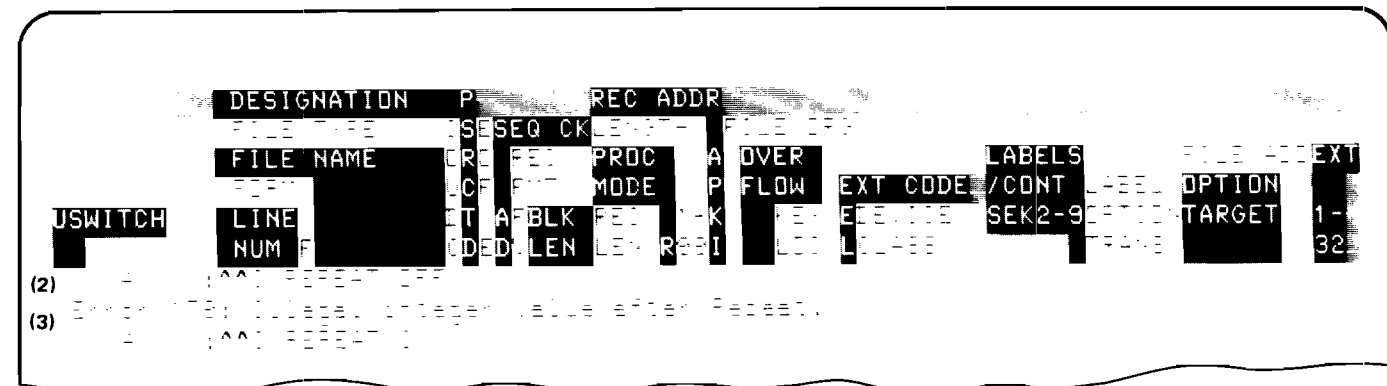
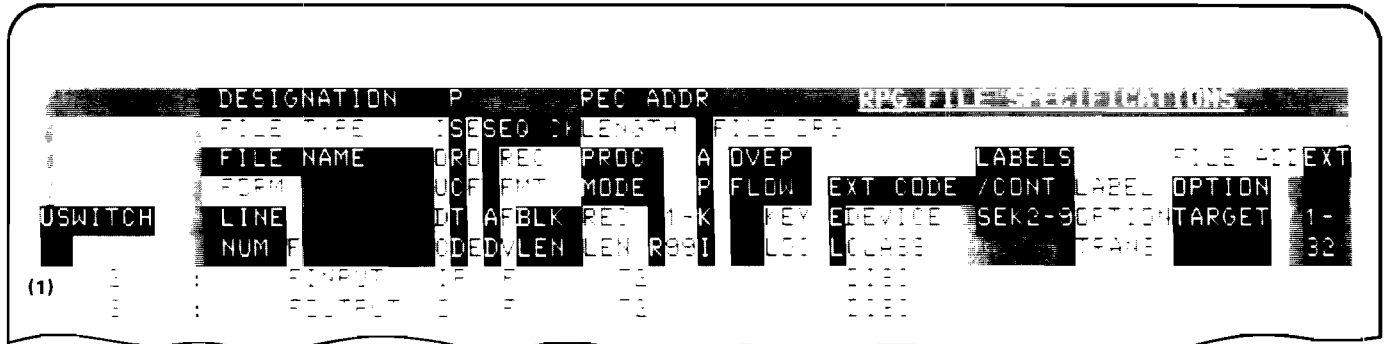
In the first example below, when the ADD command creates a work file, the ADD begins at line 1 because the file is empty. If the file existed previously, RISE would display the last line in the file and the next line number, where the ADD would begin.





In the next example, lines 2 and 3 are entered (1). With the ^R [EPEAT] 2 parameter in effect, the second screen display shows that lines 2 and 3 have cleared automatically, the cursor has homed, and line 4 has been added (2). The new line contains a syntax error in the attempt to shut off the REPEAT parameter while changing the specification form to I, for Input Specifications (3).

When the command is entered correctly, the Input Specifications form is displayed and lines 5 and 6 are added (4). Line 6 contains a command (">>") which instructs RISE to move the cursor to column 6 of the Record Specification form in order to make the entry of your specifications more convenient (5).

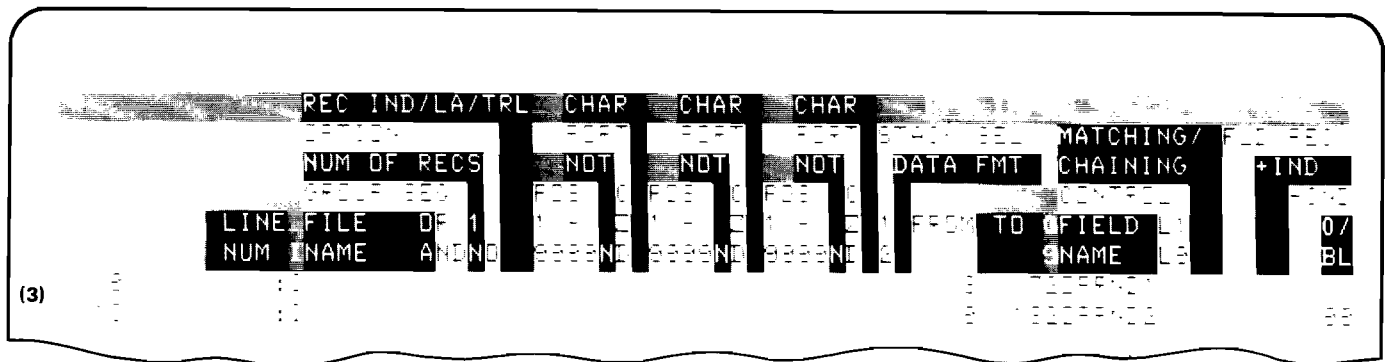
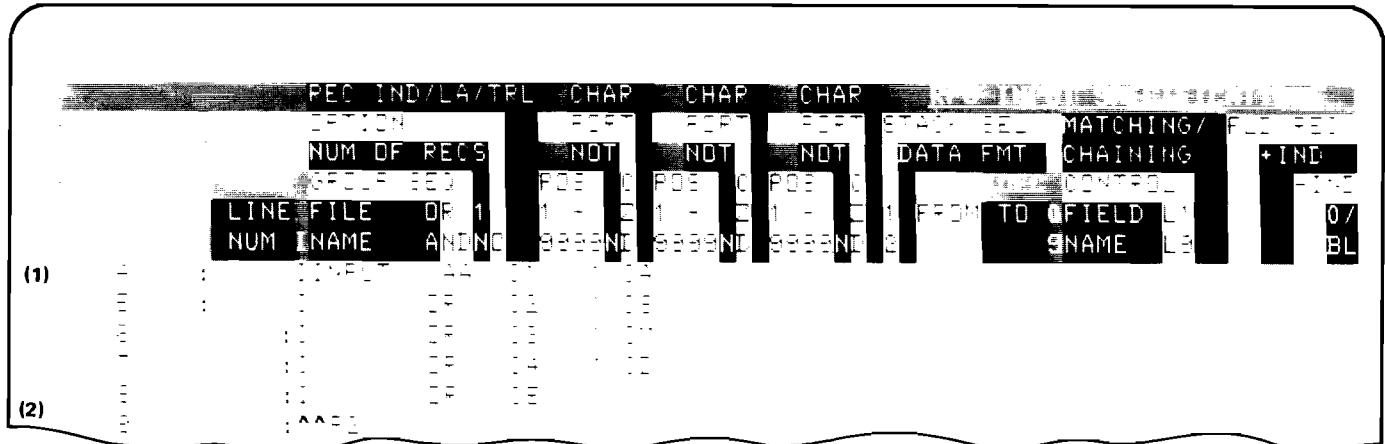


# ADD

The remaining examples show lines added correctly with the prompt at column 6 (1). In line 9, the R[EPEAT] parameter is again in effect (2). The display shows the results after two Input Specification records are entered (3). After the second line is entered (line 10), the cursor will home up and prompt for more text with line 11.

Line 11 shows the way in which you may use the L[IST] command in ADD mode if it is preceded by ">>" (4). D[ELETE] and M[ODIFY] may also be used in this mode if preceded by ">>".

<Control Y> is entered at the start of line 11 to exit ADD mode. RISE prompts you for your next command with the ">" (5).



(example continued)



# BEGIN

BEGIN allows you to initiate a new editing session as if you had just entered the RPG Interactive System Environment.

<b>Form</b>
-------------

B [EGIN] [N [OW] ]
--------------------

## Parameter explanation

N [OW] If you are editing a work file initiated with the TEXT command, NOW will cause the command to execute immediately and it will disable the safety prompts. If omitted, and you have made changes to the work file, RISE will prompt with the following message:

“KEEP not done; CLEAR current work file?”

If you respond “YES”, the command will execute and the file will be purged. A “NO” will cancel the command.

## Purpose

The BEGIN command achieves effects similar to entering an EXIT command, thus leaving RISE to end the current editing session, and then reentering RISE to begin a new editing session. If you are editing or adding to one file and you wish to clear what you have done and start over quickly, BEGIN will save time and add convenience. If the file you are editing was initiated with the FILE command, RISE will automatically close that file for you and save your changes so you can start working on a new file.

If you are editing a work file initiated by the TEXT command, the BEGIN command will clear the work file by purging it so that you may start over on a new file. Because you may wish to save the changes in a permanent file first, a safety prompt will remind you that you have not used the KEEP command with this file, and that it will be cleared upon execution. Thus you may reconsider the consequences of the command before consenting to them. With the NOW parameter in effect, however, RISE will clear the work file immediately, without displaying the safety messages.

Execution mode: Line

## EXAMPLES

The following are legal abbreviations which may be used with the BEGIN command and its parameter:

BEG            Begin a new editing session.

B N            Disable safety prompts and immediately terminate the current editing session to begin a new one.

In the first example, the ADD command is used to create a five-line file. When the BEGIN command is issued, safety prompts are returned by RISE. An affirmative response causes the command to be executed (1). The results of BEGIN are shown when LIST ALL is commanded. The execution message informs the user that there is no edit file since the file has been cleared (2). Again the ADD command is used to create a file. In this case, the new file is made permanent using the KEEP command (3). Note that the new filename, "SAVEFILE", already exists and a safety prompt appears to ensure that the user wishes to replace the old SAVEFILE with the new (4).

```
>ADD
  Creating work space file.
  1   :LINE 1
  2   :LINE 2
  3   :LINE 3
  4   :LINE 4
  5   :LINE 5
  6   :<Control Y>
>BEGIN
  KEEP not done, CLEAR current work file?YES
(1) Ready to begin new session.
>LIST ALL
(2) Error 65: There is no edit file.
>ADD
  Creating work space file.
  1   :LINE ONE
  2   :LINE TWO
  3   :LINE THREE
  4   :LINE FOUR
  5   :LINE FIVE
  6   :<Control Y>
(3) >KEEP SAVEFILE
  File exist already, purge old SAVEFILE?YES
  Keep completed.
(4) File name is SAVEFILE
```

## BEGIN

In the next example, SIMCAL is copied into a work file, and the first five lines are listed (1). When BEGIN is entered, no safety prompts appear because no changes have been made to the work file. With the ADD command, RISE is able to start a new file. Three lines are entered before <Control Y> ends Add Mode (2). The new file is then permanently stored under the filename "DATAFILE" (3).

```
>TEXT SIMCAL
  Text completed.
>LIST 1/5
  1      00011H
  SIMCAL
  2      00012FINPUT   IP   F       72           DISC
  3      00013FDOUTPUT D   F       72           DISC
  4      00014IINPUT   AA   01      1  CA
(1)  5      00015I      DR   02      1  CS
  >
  >
>BEGIN
  Ready to begin new session.
>ADD
  Creating work space file.
  1      :DATA RECORRD 1
  2      :DATA RECORRD 2
  3      :DATA RECORRD 3
(2)  4      :<Control Y>
>KEEP DATAFILE
(3)  Keep completed.
  File name is DATAFILE
  >
```

In the next example, the FILE command is used to allow for direct editing of SIMCAL (1). Since SIMCAL is not a KSAM file and must be converted to one (see FILE), an execution message is returned which prompts the user for additional information (2). The file DIRSIM is opened and 6 lines are listed (3). A CHANGE command is issued which changes the string "DISC" to "TAPE" and the results are listed (4).

Suppose at this point the user wishes to create a new file to contain important data. Currently, however, the user is directly editing DIRSIM. Before creating the new file, the user would want to close DIRSIM, making the new changes permanent and start a new editing session to create the new data file. The BEGIN command will accomplish this. When it is executed, an execution message is displayed (5) and a LIST ALL command brings an error message to the screen which indicates that RISE is ready for a new editing session (6).

```

>
(1) >FILE SIMCAL
(2) Enter new file name followed by additional number of records.
DIRSIM 10
File opened.
>LIST 1/6
1      00011H
(3)   SIMCAL
2      00012FINPUT  IP  F      72      DISC
3      00013FOUTPUT 0  F      72      DISC
4      00014IINPUT  AA  01    1  CA
5      00015I      DR  02    1  CS
6      00016I      DR  03    1  CM
(4) >CHANGE ;DISC; TO ;TAPE; IN ALL
2      00012FINPUT  IP  F      72      TAPE
3      00013FOUTPUT 0  F      72      TAPE
Changes completed.
(5) >BEGIN
Ready to begin new session.
>LIST ALL
(6) Error 65: There is no edit file.

```

# BEGIN

The user is now able to create the new data file mentioned in the last example. The ADD command is given and data records are added (1). The work file is stored to a permanent file with the KEEP command. The filename, DATAFILE, already exists, so RISE issues a safety prompt (2).

Finally, DIRSIM is again accessed for direct editing with the FILE command (3). The first 6 lines are listed (4). The listing shows that the changes were made permanent to DIRSIM when "DISC" was changed to "TAPE" (4) previously.

```
>ADD
(1)  Creating work space file.
      1      :DATA RECORD ONE
      2      :DATA RECORD TWO
      3      :DATA RECORD THREE
      4      :<Control Y>
>K DATAFILE
(2)  File exist already, purge old DATAFILE?YES
      Keep completed.
      File name is DATAFILE
      >
      >
(3)  >FILE DIRSIM
      File opened.
(4)  >L1/6
      1      00011H
      SIMCAL
      2      00012FINPUT  IP  F      72      TAPE
      3      00013FOUTPUT  O  F      72      TAPE
      4      00014IINPUT   AA  01    1  CA
      5      00015I      DR  02    1  CS
      6      00016I      DR  03    1  CM
      >
      >
      >
```



# ● CHANGE

CHANGE changes an “oldstring” to a “newstring”.

## Form

```
CH [ANGE] “oldstring” TO “newstring” [IN lb [/le] ]
```

## Parameter explanation

“oldstring” The old string you are replacing.

“newstring” The new string that is replacing the old.

● IN Indicates that a range of lines to be changed follows.

lb First, or only, line number where change will be made.

le Line where the change ends.

## Purpose

● The CHANGE command replaces “oldstring” with “newstring”. If no line range is specified, the changes will be made from the current position of the record pointer to the end of file.

## Special considerations

The number of characters in “oldstring” must be the same as that of “newstring”. If there is any difference, the column position of characters would be changed, possibly making the RPG source line syntactically incorrect.

## ● Delimiters

You may use any special characters as delimiters except the slash (/).

Execution mode: Line or Block

Record pointer: At last line changed

# CHANGE

## EXAMPLES

The following are legal abbreviations which may be used with the CHANGE command and its parameters:

CH ;OLD; TO \$NEW\$ IN F/\* Change oldstring TO newstring from first line to current position of record pointer.

CHAN;OLD;TO;NEW;IN1/10 Change oldstring to newstring in lines 1 through 10.

In the examples of the CHANGE command, a mistake is made in the spelling of the command and an error message is returned (1).

Next, there is an attempt to replace a three character string, ";PUT;" with a four character string ";FILE;" and another error message is generated (2).

When an additional space is added between the delimiters containing the oldstring ";PUT;" so that both strings are equal in length, the CHANGE is executed (3).

```
>LIST ALL
1
SIMCAL      H
2          FINPUT  IP  F      72          DISC
3          FOUTPUT  O  F      72          DISC
4
5          IINPUT  AA  01    1  CA
6          I      DP  02    1  CS
7          I      DP  03    1  CM
8          I      DP  04    1  CD
9          I      DP  05
10         I                               3  020PRND1
11         I                               9  1220PRND2          99
```

```
(1) >CHANGG ;PUT: TO ;FILE: IN ALL
      ^
```

Error 9: Syntax error on CHANGE command.

```
(2) >CHANGE ;PUT: TO ;FILE: IN ALL
```

Error 109: Oldstring length must equal newstring length.

```
(3) >CHANGE ;PUT : TO ;FILE: IN ALL
```

```
2          FINFILE  IP  F      72          DISC
3          FOUTFILE  O  F      72          DISC
5          IINFILE  AA  01    1  CA
```

# COMMENT

COMMENT produces a comment banner to contain the comments you enter.

Form COM [MENT] [C] [lb] [BY inc]
--------------------------------------

## Parameter explanation

- C** A fifteen line comment banner is displayed. (If the C parameter is not given, the default banner occupies 10 lines.) The "C" parameter displays a banner with the Column Indicator form instead of the RPG Record Specification form. A space is not required between this parameter and the command.
- lb** Location where comment banner is to be added. If this parameter is omitted, the banner is added at the end of the work file. If "lb" exists, the destination will be changed to the next sequential line.
- BY inc** Increment value for each line in the comment banner.

## Purpose

The COMMENT command allows you to enter comments in your source program wherever you wish. When the command is entered, RISE shifts you to SHOW mode and automatically provides a banner, a rectangle composed of asterisks. With it is the RPG Record Specification form corresponding to the line preceding the banner. If "C" is specified, a Column Indicator is shown with the line instead.

## Special considerations

When you wish to delete your COMMENT, you may use the DELETE command and specify the sequence range. You may also display the banner in SHOW mode and press the terminal's Clear Display key followed by the ENTER key.

Execution mode: Block

# COMMENT

## EXAMPLES

The following are legal abbreviations which can be used with the COMMENT command and its parameters:

- COMC FIRST BY 1 display fifteen line banner beginning at first line with increment value of 1.
- COM 20 Display ten line comment banner beginning at line 20.

The first example shows a ten line comment banner in Show Mode. It is displayed with the Header Specification form (1).

(1)

ENTER	INSERT	SEARCH	COMMAND	DELETE	EDIT	PRINT	END
PAUSE	LINE	OFF	WINDOW	DELETE	DELETE	DELETE	DELETE

**RPG HEADER SPECIFICATION** FILE TRANS X REF CTRL TYPE

FORM DATE FILE NO INVT BIN LOKUP	F POS	PLACE	SEQ CHECK	ERROR LG
TYPE DEBUG OPT	SEQ 1ST	ALT COL	SIGN S	DEL SKP
LINE DUMP	JREC	S,D,E	GNDR	BOSUBSUP
NUM HFILENAME1	F	ID	ITF	S
				1X1NS-V-V-V-V-V-V-V-V

PRG NAME

H\*\*\*\*\*  
H\*  
H\* THIS IS A SAMPLE COMMENT BANNER WITH AN RPG SPECIFICATION  
H\* FORM IN SHOW MODE. IT CONSISTS OF 10 LINES PER PAGE.  
H\* YOU MOVE THE CURSOR TO THE DESIRED LOCATION IN THE BANNER  
H\* AND ENTER YOUR COMMENTS.  
H\*  
H\*  
H\*  
H\*  
H\*  
H\*\*\*\*\*

COMMAND: \_\_\_\_\_  
ENTER \_\_\_\_\_  
\_\_\_\_\_



# COPY

COPY allows you to duplicate lines in a file.

<b>Form</b>
-------------

COP [Y] lb [/le] TO loc [BY inc]
----------------------------------

## Parameter explanation

- lb First — or only —line you wish to have copied.
- le Last line of a group of lines you are copying elsewhere.
- TO loc Location where the lines will be copied. The location cannot be within the range of lines specified "lb/le" after COPY. If the destination line exists, the line (s) will be copied starting at the next line in sequence.
- BY inc Increment value by which you want the lines copied. If the number of lines to be copied cannot fit from line "loc" to the next existing higher line number, the copy is not performed. Using an increment value of less than 1 allows you to concentrate more lines in a restricted location. RISE will automatically calculate a smaller increment value to concentrate more lines in a restricted area when it is possible.

## Purpose

The COPY command allows you to duplicate lines from one location to a new location in the work file. The original lines remain where they were after the COPY command is executed.

Execution mode: Line or Block

Record Pointer: At last original line that was duplicated

Control Y: Stops the listing of lines that are being copied



## EXAMPLES

The following are legal abbreviations which may be used with the COPY command and its parameters:

COP 10 TO LAST Duplicate line 10 beginning after the last line of the existing file.  
 COP 1/10TO20BY.2 Copy lines one through ten to line twenty with an increment value of .2.

In the first example, an attempt to copy lines to a location within the range of lines being copied generates an error (1).

A second error results when the range of lines to be copied is incorrectly entered (2). The third error occurs when no "TO" location is specified for the copy command (3).

Finally, a legal command causes lines 1/5 to be duplicated starting at the first line following the last line in the existing file (4).

```
(1) >COPY 1/10 TO 5
    Error 151: TO location cannot be within line range.
(2) >COPY 5/1 TO 10
    Error 82: The TO line cannot be less than the FROM line number.
(3) >COPY ALL
    Error 149: TO location missing.
(4) >COPY 1/5 TO LAST BY 1
    39      00011H
    SIMCAL
    40      00012FINPUT  IP  F      72          DISC
    41      00013FOUTPUT  O  F      72          DISC
    42      00014IINPUT   AA  01    1  CA
    43      00015I       DR  02    1  OS
    Copy completed.
>
```

# DELETE

DELETE enables you to delete lines from a work file.

## Form

```
D [ELETE] lb [/le]
```

## Parameter explanation

lb            The first or only line number where you wish to start your deletion.

le            The last line you are deleting.

## Purpose

The DELETE command is used to delete a line or range of lines from a work file. Note: when in Block Mode, you should have the sequence numbers on the screen so that you can see the sequence numbers of the lines you are deleting. Use special function key F-3 to display the numbers. Do not use the RPG line numbers to establish a line range for deletions.

## Related commands

The UNDELete command restores lines that were mistakenly deleted. Under MODIFY, the Delete subcommand interactively deletes characters within a line.

Execution mode: Line or Block

Record pointer: At the line following the last deleted line.

Control Y: Stops the listing of deleted lines. Does not stop the deletion.

## EXAMPLE

The following are legal abbreviations which may be used with the DELETE command and its parameters:

D10/50        Delete lines 10 through 50.

D10            Delete line 10.

DELE A        Delete all lines in the file.



In the example, an error is produced when an attempt to delete nonexistent lines is made (1).

A second error occurs when ALL is misspelled (2). A third results from an unnecessary character entered following last line (3).

Finally, those lines copied to the end of SIMCAL in the COPY example above are deleted by a legal command, and the results are displayed (4).

```
>
>
(1) >DELETE 200/300
    Error 81: Range is empty.
(2) >D ALLL
      ^
    Error 87: Syntax error on All option.
(3) >D F/LT
      ^
    Error 71: Syntax error on Last.
(4) >D39/LAST
    39      00011H
        SIMCAL
    40      00012FINPUT  IP  F      72          DISC
    41      00013FDOUTPUT D  F      72          DISC
    42      00014IINPUT  AA  01    1  CA
    43      00015I      DR  02    1  CS
Delete completed.
>
```

# EXIT

EXIT terminates execution of RISE.

<b>Form</b>
-------------

E [XIT] [N [OW] ]
-------------------

## Parameter explanation

**N [OW]** Exit immediately without keeping the work file and with safety prompts disabled. If you are directly editing your file, using the FILE command, the EXIT command will close the file, keeping your changes. If the file being edited is a copy of the original, initiated by the TEXT or ADD commands, the EXIT command will not terminate RISE unless a KEEP command is given to protect your recent changes. The NOW parameter allows you to EXIT immediately and discards any changes you may have made to the file.

If you choose not to use the "NOW" parameter, RISE will return a message asking if it is all right to clear the work file upon termination. If you respond with "Y [ES] ", it will clear the file and exit. If you respond with "N [O] ", it will cancel the command so that you can perform a KEEP first if you wish.

## Purpose

The EXIT command terminates the execution of RISE, returning control to MPE.

Execution mode: Line or Block

## EXAMPLE

The following are legal abbreviations which may be used with the EXIT command and its parameters:

- EXI NOW      Exit immediately without the safety prompts.
- E N          Exit immediately without the safety prompts.
- E            Exit—safety prompts may be displayed on the screen.

In the example, a misspelling of EXIT causes an error message to be displayed (1). EXIT is then entered correctly, and a safety prompt reminds the user that the KEEP command has not been entered and asks if clearing the work file is acceptable. The user responds negatively to the prompt, and the command is cancelled (2).

An unacceptable parameter — MAYBE — produces another error (3). An illegal special character is introduced between EXIT and the NOW parameter, with another error resulting (4).

Finally, EXIT NOW is entered and the command is completed, with the execution message displayed (5). The colon is MPE's prompt.

```

>
(1) >EXT
      ^
      Error 15: Syntax error on EXIT command.
(2) >EXIT
      KEEP not done, CLEAR current work file?NO
      Exit cancelled.
(3) >EXIT MAYBE
      ^
      Error 34: Only NOW parameter allowed with Exit command.
(4) >EXIT,NOW
      ^
      Error 36: Illegal special character.
(5) >EXIT NOW

      END OF PROGRAM
      :
```

# FILE

FILE allows you to edit a file directly.

## Form

```
FILE [E] filename [N [OW] ]
```

## Parameter explanation

filename      Name of the file to be edited directly.

N [OW]      Execute the command immediately without safety prompts. NOW is significant with FILE if, previous to issuing the command, you were editing a copy of another file. If you were directly editing another file, having used the FILE command, that file will be closed automatically and your changes made permanent. However, if you used the TEXT command to copy it into a work file, RISE will not open another file unless you enter the KEEP command to save your changes, or specify the NOW parameter to discard your changes. The NOW parameter is used when immediate execution is required. It disables safety prompts.

If you choose not to use the "NOW" parameter, RISE will return a safety prompt asking if you want to clear the current work file. If you respond with "Y [ES] ", it will clear the file and another file will be accessed for editing. If you respond with "N [O] ", it will cancel the FILE command so that you may perform a KEEP first if you wish.

## Purpose

The FILE command allows you to edit directly a file named "filename". If "filename" is not already a KSAM file, RISE will create one for editing and prompt you for maximum number of records in the new file. In other words, the KSAM work file is your permanent source file.

FILE is especially useful when your source program is long. If you use the TEXT/KEEP commands to bring the file to the terminal for editing or additions, this means that a permanent file must be copied into a work file (KSAM file). Such copying increases overhead. The FILE command does not require a file to be copied.

A disadvantage of the FILE command is that an error that occurs during editing cannot be corrected easily and quickly unless a backup copy was made for purposes of restoration before the editing began.

The FILE command specifies that the file "filename" is the file which is to be edited directly. RISE then ensures that "filename" is in the KSAM file form. To do so, three possible cases must be considered.

## Case 1. Creating new file

The file "filename" does not exist and must be created as a KSAM file. In order to do this, you create the file.

You type            >FILE "*filename*" \*\*

RISE responds    Creating new KSAM file, enter maximum number of records

## Case 2. Accessing old KSAM file

"filename" has been created as a KSAM file (called "newfile" below).

You type            >FILE "*newfile*"

RISE responds    File opened.

\*\* You do not enclose the filename in quotes when entering a command. Quotes are used around filename in these cases to indicate a placeholder representation of the real name of your file. For example, FILE "filename" could represent FILE SIMCAL.

## FILE

### Case 3. Converting non-KSAM file to new KSAM file

“filename” exists but is not a KSAM file. RISE must convert it. (The converted file is called “newfile” below.)

You type >FILE “*filename*”

RISE responds Enter new file name followed by additional number of records.

You type “*newfile*” 100

(If you are adding records to those in “filename” (now called “newfile”) enter the number of additional records you will add—100 in this example. The default is the number of records in “filename”.)

To access converted files in the future, type FILE “*newfile*”

#### Related commands

The TEXT and KEEP commands achieve results similar to those of FILE and EXIT. TEXT and KEEP have the disadvantage of additional overhead while files are being copied to a work file and recopied to a permanent file. The advantage is that any erroneous editing of the work file pertains only to the work file. The permanent file remains in its previous state. See TEXT and KEEP for further information concerning these commands.

It is recommended that TEXT and KEEP be used with small files since their copy time will be minimal. FILE and EXIT are best suited to large files in their developmental stage. When the source program is fully developed and in production, use TEXT and KEEP to avoid key file overhead in storing.

A KSAM file actually consists of two files: the data file and the key file. If you want to purge both parts of a KSAM file, use the following commands:

```
:RUN KSAMUTIL.PUB.SYS  
>PURGE "filename"  
>EXIT
```

For further information concerning KSAM files, please see the KSAM REFERENCE MANUAL, part number 30000-90079.

### Leaving FILE

Use the EXIT command to leave FILE. It will terminate RISE and return you to MPE with your changes made permanent. You can also use the BEGIN command to leave FILE. It will close the file you are directly editing and allow you to begin a new editing session as if you had just entered the RPG Interactive System Environment.

Execution mode: Line or Block  
Record Pointer: At first line in file

## EXAMPLES

The following are legal abbreviations which may be used with the FILE command and its parameters:

FILE FILE1 N Ready a work file named FILE1 for immediate editing.

FIL FILE1 Ready a work file named FILE1 for editing.

## FILE

In the first example, the FILE command is used to create a new KSAM file named SIMCAL2. A message is returned asking for the maximum number of records to be included in the new file. The number given is 5 (1). This is an example of Case 1 (see above).

The ADD command is entered and an attempt to add the 6th record is nullified. The error message gives the reason and recommends using the XPAND command to increase file size (2). When XPAND is entered without the number of records to expand the file by, another error message appears. (3).

The XPAND command is again tried, this time with the maximum number of records specified alphabetically rather than numerically. This generates another error message (4).

When the XPAND command is properly entered, an execution message is displayed (5). The ADD that follows is accepted by RISE and lines are added before <Control Y> is applied to exit Add mode (6).

```
>FILE SIMCAL2
(1)  Creating new Ksam file, enter maximum number of records.
      5
      File opened.
      >ADD
          1      :LINE 1
          2      :LINE 2
          3      :LINE 3
          4      :LINE 4
          5      :LINE 5
          6      :LINE 6
(2)  Error 147: No room in edit file, use XPAND.
(3)  >XPAND
      Error 137: Missing number of records to expand by.
(4)  >XPAND FIFTY
      ^
      Error 138: Illegal numeric value.
(5)  >XPAND 50
      Xpand completed.
      >ADD
          5      LINE 5
          6      :LINE 6
          7      :LINE 7
          8      :<Control Y>
(6)  >EXIT
```



In the second example, which typifies Case 3 (above), SIMCAL exists but is not a KSAM file, and RISE must convert it. A message in response to the FILE command asks for a new file name and a maximum number of records (1). When the filename and record number are given, the file is opened (2). The converted file is then listed (3).

```

>FILE SIMCAL
(1) Enter new file name followed by additional number of records.
SIMCAL3 50
(2) File opened.
(3) >L ALL
  1      00011H
SIMCAL
  2      00012FINPUT  IP  F      72          DISC
  3      00013FOUTPUT D  F      72          DISC
  4      00014IINPUT  AA  01    1 CA
  5      00015I      DR  02    1 CS
  6      00016I      DR  03    1 CM
  7      00017I      DR  04    1 CD
  8      00018I      DR  05
  9      00019I                      3  72OPRND1
 10      00020I                      9 132OPRND2          99

 11      00021C  01      OPRND1  ADD  OPRND2  RESULT 104
 12      00022C  02      OPRND1  SUB  OPRND2  RESULT
 13      00023C  03      OPRND1  MULT OPRND2  RESULT
 14      00024C  04N99  OPRND1  DIV  OPRND2  RESULT  H
 15      00025C  04 99      Z-ADD0  RESULT
 16      00026DOUTPUT  H 22      1P
 17      00027D                      11 "SIMPLE "
 18      00028D                      33 "CALCULATOR"
 19      00029D      D 11      N1P
 20      00030D
 21      00031D                      OPRND1  19 "..... YOUR PROBLEM:"
 22      00032D                      01      27 " 0 . "
 23      00033D                      02      32 "ADD "
 24      00034D                      03      32 "SUB"
 25      00035D                      04      32 "MULT"
 26      00036D                      OPRND2  32 "DIV "
 27      00037D                      39 " 0 . "
 28      00038D                      RESULT  46 "EQUALS"
 29      00039D      D 1      1P      60 " , 0 .  -"
 30      00040D      DR      N1P
 31      00041D
 32      00042D                      10 "ENTER DATA"
 33      00043D      D 11      1P      25 "IN THE FORMAT:"
 34      00044D      DR      N1P
 35      00045D
 36      00046D      D      05      13 "C NNDD NNDD"
 37      00047D
 38      00048D*END OF PROGRAM      15 "INVALID REQUEST"
>EXIT

```

## FILE

The third example of the FILE command usage represents Case 2, in which an old KSAM file (SIMCAL2, which was created in the first example) is made available for direct editing using the FILE command. RISE responds with a message stating that the file is opened (1). LIST ALL shows the contents of the file (2). Exit is used to leave RISE (3).

```
>FILE SIMCAL2
(1) File opened.
(2) >L A
      1      LINE 1
      2      LINE 2
      3      LINE 3
      4      LINE 4
      5      LINE 5
      6      LINE 6
      7      LINE 7
(3) >EXIT
```

```
END DF PRDGRAM
:
```

In the fourth example, the TEXT command is used with SIMCAL, and the first five lines are listed (1). Next, the five lines are deleted (2). When the user attempts to change work files to SIMCAL3, using the FILE command, a safety prompt is displayed. The negative response results in a cancellation of the FILE command (3).

SIMCAL, with the first five lines deleted, is then copied into a permanent file named SAVEFILE using the KEEP command (4). Two execution messages are displayed (5). A DELETE ALL command is given. Note that line 6 is the first in the file (6). <Control Y> is applied to halt the listing of the deleted lines but, as the message indicates, it does not halt the deletion (7). The FILE SIMCAL3 command, this time with the NOW parameter added to disable the safety prompts, is executed (8).

```
>TEXT SIMCAL
Text completed.
>
```

```
(1) >L 1/5
```

```
1 00011H
SIMCAL
2 00012FINPUT IP F 72 DISC
3 00013FOUTPUT O F 72 DISC
4 00014IINPUT AA 01 1 CA
5 00015I DR 02 1 CS
```

```
(2) >D 1/5
```

```
1 00011H
SIMCAL
2 00012FINPUT IP F 72 DISC
3 00013FOUTPUT O F 72 DISC
4 00014IINPUT AA 01 1 CA
5 00015I DR 02 1 CS
```

```
Delete completed.
```

```
>
```

```
>FILE SIMCAL3
```

```
(3) KEEP not done, CLEAR current work file?NO
```

```
File cancelled.
```

```
(4) >KEEP SAVEFILE
```

```
(5) Keep completed.
```

```
File name is SAVEFILE
```

```
(6) >D ALL
```

```
6 00016I DR 03 1 CM
7 00017I DR 04 1 CD
8 00018I DR 05
9 00019I 3 72OPRND1
10 00020I 9 132OPRND2 99

11 00021C 01 OPRND1 ADD OPRND2 RESULT 104
12 00022C 02 OPRND1 SUB OPRND2 RESULT
13 00023C 03 OPRND1 MULT OPRND2 RESULT
14 00024C 04N99 OPRND1 DIV OPRND2 RESULT H
15 00025C 04 99 <Control Y>
```

```
(7) Delete completed.
```

```
(8) >FILE SIMCAL3 NOW
```

```
File opened.
```

```
>EXIT
```

```
END OF PROGRAM
```

# FILE

The last example shows how to purge files created by the FILE command as well as the key files associated with KSAM files. First KSAMUTIL.PUB.SYS is run and an HP banner is displayed (1). The user then enters the file name (s) of the file (s) with the PURGE command (2). Note that the execution messages contain those filenames plus the RISE-created file designators of the associated key files in the execution messages (3).

The EXIT ends the KSAMUTIL program (4).

(1) :RUN KSAMUTIL.PUB.SYS

HP32208A.03.03 WED, MAY 27, 1981, 3:55 PM KSAMUTIL VERSION:A.03.03

HP32208A.03.03 WED, MAY 27, 1981, 3:55 PM KSAMUTIL VERSION:A.03.03

(2) >PURGE SIMCAL2

(3) SIMCAL2.RPG.SUBSYS & SIMCA97 PURGED.

>PURGE SIMCAL3

SIMCAL3.RPG.SUBSYS & SIMCA18 PURGED.

(4) >EXIT

END OF PROGRAM

:

# FIND

FIND locates the first or every occurrence of a string in a work file.

## Form

```
F [IND] [E [VERY]] "string" [IN lb [/le]]
```

## Parameter explanation

E [VERY] Every occurrence of the string is located in the given range. (If not specified, only the first occurrence is located.)

"string" Character or characters you wish to find. (Note that you may omit "string" and RISE will default and use the previous "string" given with the last FIND command. However, you must specify "string" once.)

IN Notifies RISE that the next number or character will establish the range of the search for "string".

lb First or only line designated to be searched.

le Line where the search ends.

## Purpose

The purpose of the FIND command is to locate the first occurrence of a string. If "EVERY" is specified, every occurrence of "string" will be located in the given range. If no range is given, RISE will search starting from the line after the current record pointer through to the end of the file. If "EVERY" is omitted, the command will list the first occurrence within the range. When you press "F" again, RISE will search for the next occurrence within the range. Stepping through each located line in this manner allows you to review it for any necessary modifications. When there is no occurrence of the string, RISE will print:

```
"string" not found
```

Delimiters: The delimiters around "string" may be any special character except the slash (/).

Execution mode: Line or Block

Record pointer: At the line containing "string".

# FIND

## EXAMPLES

The following are legal abbreviations which may be used with the FIND command and its parameters:

- FIN "STRING" IN A      Search the entire file for the first occurrence of "STRING". Each subsequent occurrence may be found each time "F" is pressed until all are found.
- F "STRING" IN 10/50    Find the first occurrences of "STRING" in a line range beginning with line 10 and ending with line 50.
- F EVERY "STRING" IN A    Search the entire file for all occurrences of "STRING" and list them at the terminal.

In the example, the TEXT command is used to copy SIMCAL into a work file and lines 1 through 5 are listed (1). The FIND command is used with the EVERY parameter to find the string "PUT" and all occurrences of the string are listed (2). Next, FIND is again used with EVERY but this time the string is not specified. The default string, "PUT", which was used with the last FIND command, is used again by RISE and the results are listed (3). An error is generated when an abbreviation of FIND cannot be distinguished from other commands which are similar (4). Note that "F" alone is defaulted to FIND, but when "F" is used with a second character, RISE requires a more thorough identification of the command and no default is assumed.

Finally, FIND is used with the string "DISC" in a line range. The first occurrence of "DISC" is listed on the screen (5). When the "F" command is given, the next occurrence is listed (6). An execution message informs the user that all occurrences have been found (7).

- (1) >TEXT BIMAL  
 Text completed.  
 >LIST 105  
 1 00011H  
 BIMAL  
 2 00012FINPUT IP F 72 DISC  
 3 00013FOUTPUT O F 72 DISC  
 4 00014IINPUT AA 01 1 0A  
 5 00015I DF 02 1 0B  
 >
- (2) >FIND EVERY :PUT; IN ALL  
 2 00012FINPUT IP F 72 DISC  
 3 00013FOUTPUT O F 72 DISC  
 4 00014IINPUT AA 01 1 0A  
 16 00026DOUTPUT H 22 1P  
 >
- (3) >F EV IN A  
 2 00012FINPUT IP F 72 DISC  
 3 00013FOUTPUT O F 72 DISC  
 4 00014IINPUT AA 01 1 0A  
 16 00026DOUTPUT H 22 1P  
 >
- (4) >F :DISC: IN 1..10  
 ^  
 Error 19: Can't distinguish between FILE, FIND, or FORM.  
 >
- (5) >F :DISC: IN 1..10
- (6) >F 2 00012FINPUT IP F 72 DISC
- (7) >F 3 00013FOUTPUT O F 72 DISC  
 'DISC' not found.  
 >  
 >

# FORM

FORM displays an RPG Record Specification form or column indicator on the screen.

<b>Form</b> FO [RM] form
-----------------------------

## Parameter Explanation

“form”	Meaning
H	Header specifications
F	File specifications
E	File extension specifications
L	Line counter specifications
I	Input specifications
C	Calculation specifications
O	Output specifications
CO [L]	Column indicator

Note: Space is not required between command and parameters.

## Purpose

The FORM command enables you to display an RPG Record Specification form of the type “form” in Line mode. This gives you a visual “heading” for the form with abbreviated explanations of the entries expected for the columns. A column indicator is also available with the “COL” option.

## Related Commands

While in ADD mode you can change forms by entering:

```
^^ form
```

Execution mode: Line

## EXAMPLE

The following are legal abbreviations which may be used with the FORM command and its parameters:

FORMH      Display Header Specification form.

FOO         Display Output Specification form.

FOCOL      Display column indicator



In the first example, a non-existent form is called for, using the FORM command. It generates an error message (1). The command is then entered correctly for the Column Indicator to be displayed (2).

The Column Indicator is displayed and SIMCAL, which is the work file, is listed (3). <Control Y> halts the listing (4).

```

>
>FORM XSPECS
(1)  ^
    Error 173: Only H,F,E,L,I,C,D, or CD1 allowed.
(2) >FORM COL

          0          1          2          3          4          5          6          7
(3) >L A
      1      00011H
      SIMCAL
      2      00012FINPUT  IP  F      72          DISC
      3      00013FOUTPUT D  F      72          DISC
      4      00014IINPUT  AA  01   1  CA
      5      00015I          DR  02   1  CS
      6      00016I          DR  03   1  CM
      7      00017I          DR  04   1  CD
      8      00018I          DR   05
      9      00019I                                3  720PRND1
     10     00020I                                9  1320PRND2          99

(4) <Control Y>
>

```

# FORM

In the second example, an error is generated when FORM is entered without a particular form specified (1). The FORM command is then correctly entered, with the Header Specification form specified, and the form is displayed (2). The first line of SIMCAL is listed and shown in relationship to the Header Specification form (3).

- >
- (1) >FORM  
Error 173: Only H,F,E,L,I,C,D, or CD1 allowed.
- (2) >FORMH

```

RPG HEADER SPECIFICATION
FORM DATE FILE NO INVT BIN LOKUP END SET PLACE SEQ CHECK
TYPE DEBUG OPT SEQ11ST ALT COL SIGN S DDLSIP ERROR LOG
NAME NUM HF:LENAME1 F ID S, D, E SNOB BOSUBSUP LSN 3 5 7 9 1 3 5
(3) >_ FIRST
      1 00011H
      SIMCAL
>
```

# GET

GET is used when you wish to execute a sequence of commands previously stored in a file.

## Form

```
G [ET] filename
```

## Parameter explanation

filename        The name of the file containing the commands.

## Purpose

The GET command is used when you have a number of commands stored in a file which you wish to have executed again at another time. Using "GET filename", you first have the commands displayed at the terminal. They are then executed automatically until the End of File (EOF) of the command file is reached.

The command file—the file whose commands "GET" executes—is where you imbed special specifications for the command. For example, a comment may appear in the command file by specifying an asterisk (\*) as the first character in a new line.

You may use the GET command to activate the softkeys to execute commands. The function of a particular softkey may be accomplished indirectly by putting the numbers 0 through 8 on a single line in the command file where 0=ENTER key, 1= F1 (special function key 1), 2= F2, up to 8=F8. Before entering an integer from 0 to 8, you must enter a Block mode command so that the softkeys will be activated.

You may wish to review the commands that are displayed on the screen before they are automatically executed. You do so by setting pauses to delay execution. To do this, you specify "#n" where "n" equals the number of seconds of pause you require. If "n" is not specified, the pause is turned off. It stays in effect on every command until you change it with another "#n". The maximum value of "#n" is approximately 2,147,484 seconds.

## GET

Additionally, “%” on a new line will suspend the operation of RISE until any of the special function keys is pressed while in Block Mode or until the Carriage Return key is pressed while in Line Mode. When the appropriate continuation key is pressed, RISE will continue executing the remaining commands from the command file.

If the command file contains an ADD command, RISE will enter ADD mode. Control is then returned to you so that you can enter new text as desired. When you exit ADD mode, RISE will resume control and continue to execute commands from the command file. You cannot imbed new text in the command file and expect RISE to enter it the way it executes other commands. In this respect, ADD is not automatically governed by GET as other RISE commands are.

Similarly, if the command file contains a Modify command, RISE will enter Modify mode and then return control to you. You then type in your Modify subcommands from your terminal. RISE will not read and execute subcommands stored in your command file with the Modify command.

You cannot imbed the GET command within a command file. It can only be issued at the terminal.

Note: An example of the use of the GET command is RISE TOUR. You may text in RISE TOUR.PUB.SYS and list it offline to a line printer to see an example of a command file.



## EXAMPLE


The following is a legal abbreviation which may be used with the GET command and its parameter:

G CMDFILE The RISE commands imbedded in a file named CMDFILE will be read and executed.

As an example of a file that might be used with the GET command, a brief file is created.




It is somewhat self-explanatory. Note the asterisks preceding the comments in lines 1/7 (1). In line 8, TEXT SIMCAL is entered. Following that is a command to LIST ALL (2).

The pause in line 17 is incorrectly entered and must be modified to be acceptable to RISE (3) (5).



The SHOW FIRST command is imbedded, followed by a series of 5's entered from lines 18 through 23. If the file were being run, through the use of the GET command, SHOW FIRST would bring the Header Specification form to the screen. The number 5 would operate as if special function key #5 were being pressed and you would Scroll Forward to the next Record Specification form after the pause had been completed (4). In line #26, <Control Y> is used to exit ADD mode.

MODIFY is used to insert the pound sign (#) before the pause (5). CMDFILE is then stored as a permanent file using the KEEP command (6).



# GET

```
>ADD
  Creating work space file.
  {
  (1) 1  : * THIS IS A SAMPLE COMMAND FILE.
      2  : * AN ASTERISK INDICATES A COMMENT LINE.
      3  : * YOU STORE A SEQUENCE OF COMMANDS IN A
      4  : * FILE TO HAVE THEM ALL EXECUTED
      5  : * AUTOMATICALLY.
      6  : *
      7  : * TEXT IN A FILE.
  (2) { 8  :TEXT SIMCAL
      9  :LIST ALL
     10  : *
     11  : * LET'S USE SHOW MODE TO AUTOMATICALLY
     12  : * SCROLL THROUGH THE FILE WITH THE RPG
     13  : * SPECIFICATION RECORD FORMS.
     14  : *
     15  : * WE SHOULD TURN THE PAUSE ON SO THAT WE
     16  : * HAVE TIME TO REVIEW THE SPECIFICATIONS.
  (3) 17  :20
      {
  (4) 18  :SHOW FIRST
      19  :5
      20  :5
      21  :5
      22  :5
      23  :5
      24  :EXIT NOW
      25  : * ALL DONE WITH COMMAND FILE.
      26  :<Control Y>
  }
  >
(5) >M17
     17      20
           :I#
     17      #20
           :
(6) >KEEP CMDFILE
     Keep completed.
     File name is CMDFILE
  >
  >
```

# HELP

HELP displays information concerning RISE commands.

<b>Form</b>
H [ELP] [command]

## Parameter explanation

**command** RISE command which you wish to have described in detail. You may use the abbreviation of the command so long as there is enough of it to distinguish it from other commands beginning with the same letter (s).

## Purpose

The HELP command displays information concerning RISE commands on your screen. It will also provide a detailed description of a single command which you specify. If you use HELP without specifying a single command, brief descriptions of all RISE commands are displayed at the terminal.

## Related commands

Special function key (softkey) labels are displayed on the top of the screen with the HELP command. They have the following meanings:

Key	Meaning
F1	(not used)
F2	(not used)
F3	RPG source lines with or without editor sequence line numbers will be displayed with the next SHOW command you enter
F4	move cursor to Command Window
F5	(not used)
F6	if you came from SHOW mode, return to last page that was displayed
F7	enter Line Mode
F8	display main menu

Execution mode: Block

# HELP

## EXAMPLES

The following are legal abbreviations which may be used with the HELP command and its parameters:

H L            An explanation of the LIST command will appear on the screen.

H              A brief explanation of all RISE commands will appear on the screen.

The first example shows the effects of the HELP command when the "command" parameter is not specified. Note that the portions of individual commands printed in capital letters constitute the minimum abbreviations of those commands that RISE will accept (1).

```
(1) [REDACTED] [REDACTED] SEQNUM COMMAND [REDACTED] RETURN LINE MAIN
    * OFF * WINDOW TO SHOW MODE MENU

                SUMMARY OF COMMANDS:

Add             - add new lines.
Begin           - start a new session.
Change         - change a string.
Comment       - display comment banner.
COPY          - duplicate lines.
Delete        - delete lines.
Exit          - end system.
FILE         - edit file directly.
Find         - locate a string.
FORM        - display PRP Spec form.
Get         - execute commands in a file.
help        - explain a command.
Incr       - set default increment.
Join      - append or merge a file.
Load     - save the edit file.
Line    - enter line mode.
List    - list lines.
MENU   - display main menu(bloc mode).
Modify - modify lines.
MOVE   - move lines to new location.
Print  - print lines offline.
Renum  - renumber sequence numbers or
        source line numbers.
Run    - run a program.
Show  - display a page in bloc mode.
Text  - edit a copy of a file.
Undel - undelete lines deleted by the
        last Delete command.
Verify - PRP compile or PRPF JCL or
        LIST the compilation listing.
Expand - expand the edit file size.
:      - execute MPE command.

COMMAND: [REDACTED]
ENTER
COMMAND MODE [REDACTED]
```





# INCR

INCRement resets default increment.

<b>Form</b>
I [NCR] [value]

## Parameter explanation

value            New increment value you wish to establish. (For limitations, see Note below.)

## Purpose

The INCRement command resets the default INCRement to your new "value". Once you use the INCR command, all subsequent commands which require the increment value will use your new value unless you specify a different increment in the parameters of the command. Typing "I [NCR]" without "value" resets the default value of RISE, which is 1.

Note: There are limits to the INCREMENT value that RISE will accept. The increment value must be between .001 and 5000 inclusively.

Execution mode: Line or Block

## EXAMPLES

The following are legal abbreviations which may be used with the INCRement command and its parameters:

I 100            Sets increment value to 100.

I                Resets increment value to default system value of 1.

In the first example, an error message results from an incorrect entry of the INCR command (1). When a parameter that should be numeric is entered alphabetically, a second error message appears (2). The command is then entered correctly but a listing of the file — in this case a five line truncation of SIMCAL — does not indicate any change in the increment value (3). However, when RENUMber is entered, a LIST ALL then produces the effects of the changed increment value (4).

(1) >INCC

^

Error 21: Syntax error on INCR command.

(2) >INCR FIVE

^

Error 76: Illegal numeric increment value.

>INCR 5

Increment set.

(3) >L A

1 00011H

SIMCAL

2 00012FINPUT IP F 72 DISC

3 00013FOUTPUT D F 72 DISC

4 00014IINPUT AA 01 1 CA

5 00015I DR 02 1 CS

>RENUM ALL

Renum completed.

>L ALL

5 00011H

SIMCAL

10 00012FINPUT IP F 72 DISC

15 00013FOUTPUT D F 72 DISC

20 00014IINPUT AA 01 1 CA

(4) 25 00015I DR 02 1 CS

## INCR

In the second example, the I command without a value resets the default increment value of 1 and the RE-NUMBER command is entered (1). When an illegal abbreviation of LIST is entered, an error is generated (2). A legal abbreviation of LIST ALL is then entered and the command is executed (3).

```
>I
(1) Increment reset to 1.
    >RENUM
        Renum completed.
(2) >LI AL
      ^
    Error 112: Can't distinguish between LINE or LISt command.
(3) >LIS AL
      1      00011H
      SIMCAL
      2      00012FINPUT  IP  F      72          DISC
      3      00013FDOUTPUT D  F      72          DISC
      4      00014IINPUT  AA  01    1  CA
      5      00015I      OR  02    1  CS
```

In the third example, a series of errors establishes the limits of increment values acceptable to RISE. It will not accept a value of zero (1). The minimum increment cannot be more than three digits to the right of the decimal point (2). The maximum cannot exceed 5000 (3).

```
(1) >I0
    Error 78: An increment value cannot be zero.
(2) >I.0001
      ^
    Error 74: A maximum of 3 digits is allowed right of decimal.
(3) >IN6000
    Error 156: BY increment cannot be greater than 5000.
    >
    >
```

# JOIN

JOIN merges or appends a file to your work file.

## Form

```
J [OIN] filename [TO loc] [BY inc]
```



## Parameter explanation

filename      Name of the file to be joined.

TO loc        Location where "filename" will be joined. If the destination line exists, "filename" will be joined at the next line in sequence.

BY inc        Increment value of sequence number you wish to use for the JOIN command.

## Purpose

The JOIN command appends or merges the entire contents of file "filename" to the work file. If "TO loc" is not specified "filename" is appended at the end of the work file. If "BY inc" is not specified, the current increment default value is used.

Using an increment value of less than 1 allows you to concentrate more lines in a restricted location. Sometimes the entire contents of "filename" cannot be joined as indicated because they are blocked by the next higher line number. RISE will automatically attempt to join them by using the smallest increment value of .001. If this attempt fails because of the block, the JOIN command will not execute.

Execution mode: Line or Block

Record Pointer: At last line joined to the work file

Control Y:      Stops listing of lines being joined to work file

## EXAMPLE

The following are legal abbreviations which may be used with the JOIN command and its parameters:

J FILE2 TO 10	Join the file named FILE2 to the current work file starting at line 10.
J FILE2 TO L	Join the file named FILE2 to the current work file starting at the first line following the last line in the file.
JOI FILE2 BY .1	Join the file named FILE2 at the end of the current work file using an increment value of .1.

# JOIN

In the JOIN examples, a file named RPGDOC is created for purposes of file documentation. It is stored as a permanent file using the KEEP command (1). A five-line version of SIMCAL is then copied into a work file using the TEXT command (2). The JOIN command is used to join RPGDOC to SIMCAL at line .1 (3). The results are displayed on the screen (4), and an execution message appears (5).

The LIST command, entered incorrectly, produces an error message (6). When LIST is entered legally, with a line range from the first line to line 5 (7), the display indicates how the JOIN command with an increment of .1 is executed (8). The user could JOIN RPGDOC repeatedly to different source files to document those files.

```

      0          1          2          3          4          5          6          7
1234567890123456789012345678901234567890123456789012345678901234567890
  1   :      H*
  2   :      H* PROGRAM NAME:
  3   :      H*
  4   :      H* PROGRAMMER  :
  5   :      H* DATE WRITTEN:
  6   :      H* PURPOSE    :
  7   :      H*
  8   :<Control Y>
(1) >K RPGDOC
    Keep completed.
    File name is RPGDOC
    >
    >
    >
    >T SIMCAL
    Text completed.
(2) >L 1/5
    1      00011H
    SIMCAL
    2      00012FINPUT  IP  F      72          DISC
    3      00013FOUTPUT D  F      72          DISC
    4      00014IINPUT  AA  01    1  CA
    5      00015I      DR   02    1  CS
(3) >JOIN RPGDOC TO .1
    .1      H*
    .2      H* PROGRAM NAME:
    .3      H*
    .4      H* PROGRAMMER  :
    .5      H* DATE WRITTEN:
    .6      H* PURPOSE    :
    .7      H*
(4) Join completed.
(5) >
```

(example continued)

(6) >LF/5

Error 112: Can't distinguish between LINE or LIST command.

(7) >L F/5

```

.1      H*
.2      H* PROGRAM NAME:
.3      H*
.4      H* PROGRAMMER  :
.5      H* DATE WRITTEN:
.6      H* PURPOSE    :
.7      H*

```

1 00011H

SIMCAL

```

2 00012F INPUT IP F 72 DISC
3 00013F OUTPUT O F 72 DISC
4 00014I INPUT AA 01 1 CA
5 00015I OR 02 1 CS

```

(8)

>

# KEEP

KEEP saves the contents of the work file in a permanent file.

## Form

```
K [EEP] [filename] [N [OW]] [U [NN]]
```

## Parameter explanation

- filename      Name of the permanent file where you wish to copy the work file.
- N [OW]        Execute the command immediately, disabling safety prompts and overwriting an old permanent file with the same name if it exists. (Note: "filename" cannot be "N", "NO", "NOW", "U", "UN", or "UNN". These letters are reserved for literal parameters in the command.)
- U [NN]        KEEP the file unnumbered (without the appended sequence numbers). If you omit "UNN", the sequence numbers will be stored in columns 81-88 of the permanent file.

## Purpose

The KEEP command saves the contents of the work file in a permanent file designated "filename". If "filename" is omitted, the file will be kept under the current work file name which was designated previously with the TEXT command. If "filename" does not exist, RISE will create that file.

If "filename" exists, RISE will not immediately overwrite the existing file unless you use the NOW parameter. Instead, it will return a message informing you that "filename" already exists, and asking if you wish to overwrite the existing permanent file. You respond with "Y [ES]" to replace it or "N [O]" to cancel the KEEP command.



The "NOW" option ensures that the overwriting will take place immediately. It also disables the safety prompts. It saves time but it should only be used when you are certain that the work file should replace the permanent file. There is no provision for reconsidering changes once NOW is used. When you omit NOW, and respond to the message informing you that the "filename" exists and asking if you wish to overwrite the existing copy, you do have an opportunity to reconsider the replacement. However, there is a slight delay involved.

### Related commands

A file which you KEEP may be copied back to the work file using the TEXT command so that you can edit or add to it. To TEXT and KEEP a large RPG source file each time you have changes to make can entail long delays. You will save time by editing a long file directly using the FILE/EXIT commands. See the discussion of FILE for advantages and disadvantages of using the FILE/EXIT commands.

Execution mode: Line or Block

## EXAMPLE

The following are legal abbreviations which may be used with the KEEP command and its parameters:

K FILE1 N	Copy the file named FILE1 into a permanent file. If FILE1 exists, overwrite the old copy with the contents of the current work file without displaying safety prompts.
K NEWFILE	Copy the file named NEWFILE into a permanent file.
K N U	Copy the current work file into a permanent file immediately without appended sequence numbers.

## KEEP

In the first example, SIMCAL is copied into the work file with the TEXT command. It is then copied to a permanent file called BACKUP using the KEEP command, and the execution messages are displayed (1). When lines 1 through 5 of the work file are deleted, and an attempt to KEEP BACKUP UNNumbered is entered, safety prompts appear (2), giving the user a chance to reconsider the effects of overwriting the original copy of BACKUP with the edited copy now in the work file. When the prompt is answered in the affirmative, the KEEP command is executed (3).

Line 38 is then modified, and KEEP BACKUP is entered with the NOW and UNN parameters (4). NOW disables the safety prompts and the command is executed immediately (5).

```
>T SIMCAL
  Text completed.
>KEEP BACKUP
  Keep completed.
(1)  File name is BACKUP
      >
      >DELETE1/5
          1      00011H
          SIMCAL
          2      00012FINPUT   IP  F      72          DISC
          3      00013FDOUTPUT D  F      72          DISC
          4      00014IINPUT   AA  01    1  CA
          5      00015I       DR   02    1  CS
      Delete completed.
(2)  >KEEP BACKUP UNN
      File exist already, purge old BACKUP?YES
      Keep completed.
(3)  File name is BACKUP
      >
      >MODIFY LAST
          38      000480*END OF PROGRAM
                   :
                   RNAMED SIMCAL
          38      000480*END OF PROGRAM NAMED SIMCAL
                   :
(4)  >KEEP BACKUP NOW UNN
      Keep completed.
(5)  File name is BACKUP
      >
```

In the second example, a file named TESTFILE is copied into the work file. TESTFILE was previously stored as a permanent file with the UNNumbered parameter. Because sequence numbers are essential as references in editing, they are appended automatically for the user's convenience (1). The CHANGE command is used on TESTFILE, and an execution message informs the user that all the changes have been made (2).

Entering EXIT causes safety prompts to appear because, if the EXIT is executed without a preceding KEEP command, the changes made to the work file would not be saved. When the prompts are answered negatively, the command is cancelled (3). KEEP is then tried, and the safety prompts displayed (4). This time the user indicates that the permanent copy of TESTFILE should be overwritten by the edited work file. The user enters the EXIT command, and this time it is safe to exit since the changes were kept in a permanent file first (5).

```

>
>TEXT TESTFILE
(1) File is unnumbered, appending sequence numbers.
Text completed.
>CHANGE ;RESULT: TD ;OUTPUT: IN ALL
   6  000210  01  OPRND1  ADD  OPRND2  OUTPUT 104
   7  000220  02  OPRND1  SUB  OPRND2  OUTPUT
   8  000230  03  OPRND1  MULT OPRND2  OUTPUT
   9  000240  04N99  OPRND1  DIV  OPRND2  OUTPUT
  10  000250  04 99      Z-ADD0
  23  000380
(2) Changes completed.
>EXIT
KEEP not done. CLEAR current work file?NO
(3) Exit cancelled.
>KEEP
File exist already, purge old TESTFILE?EE
(4) keep completed.
File name is TESTFILE
(5) >EXIT

```

# LINE

LINE places you in Line Mode.

<b>Form</b>
LIN [E]

## Purpose

This command places you in Line Mode. There are many advantages to using LINE Mode, among them convenience and speed. The LINE command allows you to enter commands one at a time by pressing the carriage return. Errors can be erased with the backspace key.

In Line Mode you see the results of commands being executed on the screen. For example, if you use the DELETE command, the lines being deleted are displayed. (You may not wish to have all the lines being deleted displayed. By typing <Control Y> you stop the listing. <Control Y> does not stop the execution of the command, however.)

## Related commands

Another method of entering Line Mode from Block Mode is the the special function key F7, when the softkey label for F7 reads "LINE MODE".

Execution mode: Line

## EXAMPLES

The following is a legal abbreviation of the LINE command:

LIN            Shift to Line Mode.

# LIST

LIST prints the work file on the screen or on the line printer.

## Form

```
L [IST] [lb [/le]] [O [FF]]
```

## Parameter explanation

- lb            First or only line you want to have displayed.
- le            Last line of a range you want displayed on the screen.
- O [FF]        Print offline on a line printer with the formal file name of "RISELIST" and device class of "LP".

## Purpose

The LIST command prints all or a portion of the work file on the screen or line printer. If you do not specify a line or lines for listing, only the line following the current position of the record pointer is listed. You specify the "OFF" parameter to list offline to a line printer.

## Related Command

The PRINT command will also list lines offline on a line printer. PRINT has an additional feature: whenever the RPG form type in column 6 changes, it skips three lines so that your listing is in a more readable form. The PRINT command will also print offline the compilation listing produced by a V [ERIFY] R [PG] command.

Execution mode: Line  
Record pointer: At the last line printed  
Control Y: Halts execution of the command

# LIST

## EXAMPLE

The following are legal abbreviations which may be used with the LIST command and its parameters:

- LIS A List the complete contents of the work file.
- L F/10 List the first through line 10.
- L \*-5/\*+5 List a range of lines beginning five lines before the current position of the record pointer and ending five lines after the record pointer.
- L 10 List line 10.
- L List the line following the position of the current record pointer. (Used for stepping through a file, line by line.)

In the first example, TEXT is used to copy SIMCAL into a work file and the first five lines are listed on the screen using the LIST command (1). The LIST command is then used with the asterisk — (\*) — which symbolizes the current position of the record pointer. The command calls for a range of ten lines beginning five lines before the current record pointer position. Since the previous command left the record pointer at line 5, when the command is executed, lines 1 through 10 are displayed (2).

When the L [IST] command is used without any parameters, the line following the current position of the record pointer is listed. Thus, lines 11 through 13 are stepped through (3).

```
(1) >T SIMCAL
    Text completed.
```

```
>L 1/5
```

```
 1      00011H
SIMCAL
 2      00012FINPUT  IP  F      72          DISC
 3      00013FOUTPUT  D  F      72          DISC
 4      00014IINPUT   AA  01    1  CA
 5      00015I        DR  02    1  CS
```

```
(2) >L*-5/**+5
```

```
 1      00011H
SIMCAL
 2      00012FINPUT  IP  F      72          DISC
 3      00013FOUTPUT  D  F      72          DISC
 4      00014IINPUT   AA  01    1  CA
 5      00015I        DR  02    1  CS
 6      00016I        DR  03    1  CM
 7      00017I        DR  04    1  CD
 8      00018I        DR  05
 9      00019I                                3  72OPRND1
10     00020I                                9 132OPRND2          99
```

```
(3) { >L
      11     00021C  01          OPRND1  SUB  OPRND2  RESULT  104
      >L
      12     00022C  02          OPRND1  SUB  OPRND2  RESULT
      >L
      13     00023C  03          OPRND1  MULT OPRND2  RESULT
```

## LIST

In the second example, LIST is used to bring the last line of the file to the screen (1). Next, the special character "@", which has the same meaning to RISE as the asterisk (\*), is used with LIST to display the five lines preceding the current position of the record pointer — at line 38 after the last command. Thus, the last five lines of the file are listed (2).

```
(1) >L LAST
    38      000480*END OF PROGRAM

(2) >L @-5/L
    33      000430          D 11      1P
    34      000440          DR          N1P
    35      000450                                13 "C NNNDD NNNDD"
    36      000460          D          05
    37      000470                                15 "INVALID REQUEST"
    38      000480*END OF PROGRAM
```

In the next example, an error results when non-existent lines are used with the LIST command (1). Following that, LIST is entered with a "line beginning" which is higher in sequence than the "line ending" and another error message is shown (2). Next, the same error is committed using special characters to indicate the current position of the record pointer (3). LIST used with a line number which includes an illegal decimal value causes another error (4). Finally, an illegal character used with the FIRST parameter brings up another error message.

```
(1) >L 100/150
    Error 81: Range is empty.

(2) >L 10/5
    Error 62: The TO line cannot be less than the FROM line number.

(3) >L @+5/@-5
    Error 62: The TO line cannot be less than the FROM line number.

(4) >L 10.0005
    Error 74: A maximum of 3 digits is allowed right of decimal.
    >

(5) >L FIRST/
    Error 61: Only a '*',line number,FIRST,LAST,ALL allowed in range.
    >
```



# MENU

MENU displays the main menu.

Form
ME [NU]

## Purpose

The MENU command displays the main menu. It contains four sets of commands and the labels of the special function keys. The command sets are segregated by function: File Commands, Editing Commands, General-1 Commands, and General-2 Commands. See the example of the menu below for the commands which are associated with each heading.

## Related commands

To enter a command, you press the special function key which corresponds to the set containing the commands you wish to use. The softkey moves the cursor to the first inverse video (white) box. You type in the command abbreviation. The cursor then moves to the next box, where you enter the parameters of the command. Under each box is a description of the appropriate parameter for that space. After the menu is filled out, you press the ENTER key to execute the command. (Note that not all RISE commands appear on the menu. Only the most important do.)

When the main menu is displayed, the special function keys perform the following operations:

- F1: move cursor to File Commands
- F2: move cursor to General-1 Commands
- F3: show RPG source lines with or without editor sequence numbers when you enter the next SHOW command
- F4: move cursor to Command Window
- F5: move cursor to Edit Commands
- F6: move cursor to General-2 Commands
- F7: enter Line Mode
- F8: refresh main menu (clears the screen and redraws the menu)

Execution mode: Block

# MENU

## EXAMPLES

The following is a legal abbreviation which may be used for the MENU command:

ME Display the main comand menu.

The first example shows the main menu without any commands entered in any of the unprotected fields (1).

```
(1)  FILE  GEN1  SEQNUM  COMMAND  EDIT  GEN2  LINE  REFRESH
      CMDS  CMDS  * OFF *  WINDOW  CMDS  CMDS  MODE  MENU

PB: FILE COMMAND                      PB: GENERAL COMMANDS
█ - █ - █ - █ - █                      █ - █ - █ - █ - █
File filename ->QW                      Inch value
Text filename ->QW                      Print from ln - to line - PPG LIST
Head filename ->QW                      Com ->QW                      at line - by and
Join filename - to line - by and       Renum from ln - to line - by and
                                        Move - from ln - to line - at line - by and

PB: EDITING COMMANDS                  PB: GENERAL COMMANDS
█ - █ - █ - █ - █                      █ - █ - █ - █ - █
Delete - from line- to line
Show - from line- to line
Add - at line - by and
Find - string - from line- to line
Loc - from line- to line - at line - by and
Change - oldstring- newstring- from ln - to line

COMMAND: [ █ ]
EXIT
COMMAND MODE
```

The second example shows the menu with the TEXT command selected in the upper left corner. Two parameters are also entered to complete the full command of "TEXT SIMCAL NOW" (2).

```

(2) FILE CMDS   GEN1 CMDS   SEQNUM * OFF *   COMMAND WINDOW   EDIT CMDS   GEN2 CMDS   LINE MODE   REFRESH MENU

F1: FILE COMMANDS                               F2: GENERAL1 COMMANDS
- SIMCAL -NOW - - - - - - - - - - - - - - -
File-filename -NOW                               Inch-value
Text-filename -NOW                               Print-from ln - to line - PPG LIST
Keep-filename -NOW -UNN                         Com -C - at line - by inc
Join-filename -to line -by inc                   Penum-from ln - to line - by inc
                                                Move -from ln - to line - at line - by inc

F3: EDITING COMMANDS                             F4: GENERAL2 COMMANDS
- - - - - - - - - - - - - - - - - - - - -
Delete- from line- to line                       Undel
Show - from line- COL ind                         Help - command
Add - at line - by inc                           Verify- PPG or LIST
Find - string - from line- to line               Exit - NOW
Copy - from line- to line - at line - by inc
Change- oldstring- newstring- from ln - to line

COMMAND: [ ]
ENTER?
COMMAND MODE
  
```

# MODIFY

MODIFY enables you to modify lines interactively.

## Form

```
M [MODIFY] [lb [/le]] [^^]
```

## Parameter explanation

- lb            Single line or first line of range you wish to modify.
- le            Last line of range you intend to modify.
- ^^            Displays the line or lines for modification with the corresponding RPG Record Specification form.

## Purpose

The MODIFY command is your means of modifying lines interactively. It operates similarly to the same command in Edit/3000. The line which is selected for modification is displayed at the terminal. You modify the line by positioning the cursor beneath the first, or only, character you wish to modify. Three subcommands operate with MODIFY. They are insert (I), replace (R), and delete (D).

## Related Commands

### *Insert ("I") subcommand*

"Istring" will insert a character or string. You position the cursor beneath the position where you wish the insertion to begin. You type an "I" immediately followed by the "string" to be inserted.

***Replace ("R") subcommand***

"Rstring" will replace the characters in the line marked with "R" by your new "string". Again, you position the cursor beneath the first character you wish to replace before typing "Rstring".

***Delete ("D") subcommand***

To delete a character or string of characters, type "D" beneath the first character to be deleted. A range of "D" 's will remove the string between the first and last "D". If there is more than one "D" used in modifying a single line, all characters between the leftmost and rightmost "D" will be deleted.

You can use "I" (insert) and "R" (replace) with a "D" or range of "D" 's. If only one "D" is used, followed by an "I" or "R", the second "D" is inferred to be the last character before the "I" or "R". If more than one "D" is used, the characters between the first and last will be deleted, and the second subcommand will be executed as usual at the character marked by the "I" or "R".

***Undesignated subcommand***

When you use the MODIFY command, the line designated appears on the screen. If you do not indicate the subcommand you are using (D,I, or R), whatever is typed beneath the line to be modified will replace the character or characters directly above unless blanks alone are typed. In other words, R (eplace) is the default subcommand.

# MODIFY

## Leaving MODIFY

If you specify a range of lines to be modified and wish to leave the mode before going through the full range, type double slash (//) at the beginning of the next line. When you are modifying a single line, press the carriage return at the beginning of a new line after the source line appears on the screen in the form desired.

## Control Y

If you do not wish to make the modifications you originally thought were necessary, type <Control Y> before you press the carriage return at the start of a new line. (Pressing the carriage return at the start of a new line means leaving modification mode for this line and permanently storing the newly modified line.) RISE will respond with the message "Restore Record", and the changes will not be made.

The UNDELeTe command does not work with deletions made in the the MODIFY mode.

Execution mode: Line  
Record pointer: At last line modified

## EXAMPLE

The following are legal abbreviations which may be used with the MODIFY command and its parameters:

M */LAST^^	Modify lines from the current position of the record pointer to the last line and display the corresponding RPG Record Specification form.
M LAST	Modify the last line of the file.
M ^^	Modify line at the current position of the record pointer and display the corresponding RPG Record Specification form.
M	Modify line at the current position of the record pointer.
MOD AL^^	Modify all the lines and display the corresponding RPG Record Specification form.

The first five lines of SIMCAL are used for these examples. After the lines are listed, MODIFY FIRST is entered and a two-character string is replaced using the R subcommand (1). Note that the line selected for modification is displayed unmodified on the screen initially, and that it is displayed again after the execution of the subcommand. An effort to use an illegal parameter with MODIFY evokes an error, and the suitable message is displayed (2).

Line 2 is singled out for a number of modifications. Initially, the word INPUT is deleted. Note that the D's are placed directly beneath the first and last characters of the string INPUT and that this is sufficient to delete the complete word (3). <Control Y> is used to demonstrate a method of restoring an erroneous deletion (4).

```

>L 1/5
  1  00011H
SIMCAL
  2  00012FINPUT  IP  F      72          DISC
  3  00013FDUTPUT D   F      72          DISC
  4  00014IINPUT  AA  01    1  CA
  5  00015I      DR   02    1  CS
(1) >MOD FIRST
    1  00011H
SIMCAL
      :  R10
    1  00010H
SIMCAL
      :
(2) >M SECDND
      ^
      Error 61: Only a '*',line number,FIRST,LAST,ALL allowed in range.
(3) >M 2
    2  00012FINPUT  IP  F      72          DISC
      :      D  D
(4)  2  00012F  IP  F      72          DISC
      :<Control Y> Restore Record.

```

## MODIFY

In the next example, the delete subcommand removes a three-character string from the word INPUT (1). Note that the remainder of the line moves left a corresponding number of spaces (2). Thus, the syntax of an RPG source program will be affected by modifications which remove characters or insert characters into records. Next, the string removed by the delete subcommand is replaced by a three character insertion (3).

The replace (R) subcommand is used next (4). Though two more characters are added to the line, they replace blank spaces, and the syntax of the line is not changed. Replace (R) is used again on the next line (5). And the word DISC is replaced by TAPE in the line following, although the subcommand is not used. This is an example of an undesignated subcommand (6).

```
(1) 2      00012FINPUT  IP  F      72          DISC
      :           D D
(2) 2      00012FIN   IP  F      72          DISC
(3) 2      :           IFIL
(4) 2      00012FINFIL IP  F      72          DISC
      :           RES
(5) 2      00012FINFILES IP  F      72          DISC
      :           R80
(6) 2      00012FINFILES IP  F      80          DISC
      :           TAPE
```

In the next example, the delete (D) subcommand is used in conjunction with the insert (I) subcommand with the result that "MAGTAPE" replaces "TAPE" in the line (1). The next example demonstrates how all characters between the two outside D's used in the delete subcommand are removed upon execution of the subcommand (2). Again a <Control Y> is used to restore the characters erroneously deleted (3). Finally, the delete and insert subcommands are again used together, and the results are shown (4).

```
(1) 2      00012FINFILES IP  F      80          TAPE
      :           D DIMAGTAPE
(2) 2      00012FINFILES IP  F      80          MAGTAPE
      :D DDD          D
(3) 2      80          MAGTAPE
      :<Control Y> Restore Record.
(4) 2      00012FINPUT  IP  F      72          DISC
      :DDDDDI12345
      2      12345FINPUT  IP  F      72          DISC
      :
>
```



In the final example of the MODIFY command, MODIFY ALL is entered with the additional parameter calling for the record specification form corresponding to the line being modified (1). The first line, then, is shown beneath the Header Specification form (2). It is modified by spacing over and typing a "B" as shown under the form (2). After typing the "B", the user hits the carriage return and RISE re-displays the modified line (3). Lines 2 and 3 are then displayed under the RPG Specification form (4) (5). A double slash (//) entered at the beginning of line 3 ends the modification mode (6).

(1) >MODIFY ALL \*\*

(2)

RPG HEADER SPECIFICATION										FILE TRANS	X REF	CTLPL TYPE
FORM	DATE	FILE	NO	INVT	BIN	LOKUP	F POS	PLACE	SEQ CHECK			
TYPE	DEBUG	OPT	SEQ11ST	ALT	COL	SIGN	S	DOLSP	ERROR	NUMBER		
PRG	LINE	DUMP	JREC	S,D,E	SMIOB	BOSUB	SUP	LSN	3	5	7	9
NAME	NUM	HFILENAME	F	ID		1TF	S	1X1NS	=	=	=	=
	00010H											
EIMCAL	:											

(3)

RPG HEADER SPECIFICATION										FILE TRANS	X REF	CTLPL TYPE
FORM	DATE	FILE	NO	INVT	BIN	LOKUP	F POS	PLACE	SEQ CHECK			
TYPE	DEBUG	OPT	SEQ11ST	ALT	COL	SIGN	S	DOLSP	ERROR	NUMBER		
PRG	LINE	DUMP	JREC	S,D,E	SMIOB	BOSUB	SUP	LSN	3	5	7	9
NAME	NUM	HFILENAME	F	ID		1TF	S	1X1NS	=	=	=	=
	00010H											
EIMCAL	:											

(4)

RPG FILE SPECIFICATIONS									
DESIGNATION	P	REC	ADDR						
FILE TYPE	ISE	SE	SE	SE	SE	SE	SE	SE	SE
FILE NAME	OP	REC	PP	OC	A	OVER		LABELS	FILE ADDEXT
FORM	UCF	FMT	MODE	P	FLOW	EXT CODE	/	CONT	LABEL
USWITCH	LINE	DT	AF	BL	REC	1-K	KEY	E	DEVICE
	NUM	CO	ED	VLEN	LEN	R99I	LOC	L	CLASS
	3	13345F	INPUT	IP	F	72			DISC
	:								

(5)

RPG FILE SPECIFICATIONS									
DESIGNATION	P	REC	ADDR						
FILE TYPE	ISE	SE	SE	SE	SE	SE	SE	SE	SE
FILE NAME	OP	REC	PP	OC	A	OVER		LABELS	FILE ADDEXT
FORM	UCF	FMT	MODE	P	FLOW	EXT CODE	/	CONT	LABEL
USWITCH	LINE	DT	AF	BL	REC	1-K	KEY	E	DEVICE
	NUM	CO	ED	VLEN	LEN	R99I	LOC	L	CLASS
	3	00013F	OUTPUT	O	F	72			DISC
	:								

(6) >

# MOVE

MOVE transfers lines to a new location.

## Form

```
MOV [E] lb [/le] TO loc [BY inc]
```

## Parameter explanation

- lb**                Single line you wish to move or the beginning of a range of lines you are moving.
- le**                Last line of range you wish to move.
- TO loc**           Location where the lines are to appear. If destination line exists, the line (s) will be moved starting at the next line in sequence.
- BY inc**           Increment of line numbers you wish to use. If "BY inc" isn't specified, the current default increment is used. If you attempt to move too many lines into a restricted position, the move may be blocked by the next higher line number. When this occurs, none of the lines are moved.

## Purpose

The MOVE command transfers a line or lines to a designated location.

## Related commands

The COPY command duplicates a line or lines in a new position while leaving the original lines where they are. The MOVE command does not leave lines in their original position; they are physically transferred.

Execution mode: Line or Block

Record pointer: At the line following the last line that was moved

Control Y: Stops the listing of lines being moved

## EXAMPLES

The following are legal abbreviations which may be used with the MOVE command and its parameters:

MOV 10 TO 20                    Move line 10 to line 20. (If data has been entered on line 20, RISE will move 10 to the next higher line by automatically creating an increment such as 20.1 if that is possible.)

MOV \*-5/\*+5 to \*+20           Move a range of lines beginning 5 lines before the current record pointer to a position 20 lines after the current record pointer.

MOV F TO LAST                   Move the first line to the last line in the file.

In the first example, SIMCAL is copied into a work file using the TEXT command and five lines are brought to the screen using LIST (1). The MOVE command is then used to move line 2 to line 3.5 (2). The results are automatically displayed (3). Lines 1 through 5 are then LISTed (4).

```

>T SIMCAL
Text completed.
(1) >L 1/5
1      00011H
SIMCAL
2      00012FINPUT  IP  F      72          DISC
3      00013FDOUTPUT D  F      72          DISC
4      00014IINPUT  AA  01    1 CA
5      00015I      DR  02    1 CS
(2) >MOVE 2 TO 3.5
3.5    00012FINPUT  IP  F      72          DISC
(3) Move completed.
(4) >L1/5
1      00011H
SIMCAL
3      00013FDOUTPUT D  F      72          DISC
3.5    00012FINPUT  IP  F      72          DISC
4      00014IINPUT  AA  01    1 CA
5      00015I      DR  02    1 CS

```

# MOVE

In the next example, the record pointer is moved to the last line in the file when LIST LAST is entered (1). A command is then given to MOVE a range of lines beginning ten lines before the current record pointer and ending with the last line in the file to line 100 (2). The results appear on the screen (3). An attempt to move an empty line range brings up an error message (4). Another error is committed when a MOVE command is entered with only a portion of the line range specified (5). Finally, a third error is caused by a MOVE command used with an illegal destination value (6).

```
(1) >L LAST
    38      000480*END OF PROGRAM
(2) >MOVE *-10/LAST TO 100
    100     000380                      RESULT  60 "   , 0 .   -"
    101     000390          D 1          1P
    102     000400          DR          N1P
    103     000410
    104     000420                      10 "ENTER DATA"
    105     000430          D 11         1P
    106     000440          DR          N1P
    107     000450
    108     000460          D           05
    109     000470                      13 "C NNNDD NNNDD"
(3) 110     000480*END OF PROGRAM
      Move completed.
(4) >MOVE 50/70 TO FIRST BY .11
      Error 81: Range is empty.
(5) >MOVE FIRST
      Error 149: TO location missing.
(6) >MOVE FIRST TO .1155
      Error 74: A maximum of 3 digits is allowed right of decimal.
>
```

# PRINT

PRINT enables you to print a complete work file or portion of the file, or a compilation listing of a file, on the line printer.

```
Form
P [RINT] lb [/le] [R [PGLIST]]
```

## Parameter explanation

- lb First, or single, line you wish to have printed.
- le Last line of a range you want printed offline.
- RPGLIST Print the last RPG compilation listing produced by the RISE VERIFY command.

## Purpose

The PRINT command prints a line, a line range, or a complete file on the line printer in a special format with the formal file name of "RISELIST" and device class of "LP". Whenever the RPG form type in column 6 changes, the PRINT command will skip three lines to group together those lines which have the same form type. This produces a more readable format for the source code. Because of this feature, the PRINT command should only be used when you are working on RPG source code. If "RPGLIST" is specified, the compilation listing produced by the VERIFY RPG command is printed instead of the work file. In this case, the formal file name used is "RPGLIST".

## Related commands

The LIST command will also print your workfile offline on a line printer. It does not skip 3 lines when column 6 changes. Thus, it is suitable for printing offline any type of text you may be editing, such as data or a document.

Execution mode: Line or Block

# PRINT

## EXAMPLES

The following are legal abbreviations which may be used with the PRINT command and its parameters:

P FIRST/50	Print a range of lines from the beginning to line 50.
P ALL RPGLIST	Print the entire listing produced by the VERIFY RPG command.
PRI A R	Print the entire listing produced by the VERIFY RPG command.
P*-5/L	Print a range of lines beginning five lines before the current position of the record pointer and ending with the last line in the file.

In the example, the TEXT command is used to copy SIMCAL into a work file. PRINT ALL is entered and an execution message appears on the screen when the command is executed (1). An error results when PRINT is used with the RPGLIST parameter and SIMCAL has not been previously compiled using the VERIFY RPG command (2). Finally, an abbreviation of the PRINT command is shown to be legal by the appearance of the execution message (3).

```
>T SIMCAL
  Text completed.
(1) >PRINT ALL
     Print completed.
     >P ALL RPGLIST
(2)  Error 64: There is no compilation listing to print.
     >
     >PRIN1/10
(3)  Print completed.
     >
     >
```

# RENUM

RENUM rennumbers sequence line numbers.

Form			
	[lb/le]	[BY inc]	
	BOTH	[BY inc]	
R [ENUM]	SO [URCE]	[BY inc]	
	SO [URCE]	W [ITH]	SE [QNUM]
	SE [QNUM]	W [ITH]	SO [URCE]

## Parameter explanation

### *Parameter option 1:* [lb/le] [BY inc]

Rennumbers the range "lb/le" by "inc". If the line range is omitted, the entire file will be renumbered. ( As in other commands, "ALL" may be used in place of "lb/le".) If "BY inc" is omitted, the default increment is used.

### *Parameter option 2:* BOTH [BY inc]

Rennumbers both the editor sequence numbers and the RPG source line numbers (columns 1-5) by "inc". Both will have the same value. The value of "inc" cannot be less than 1. Using this option you renumber your entire file.

### *Parameter option 3:* SO[URCE] [BY inc]

Rennumbers the RPG Source program line numbers (col. 1-5). If "BY inc" is omitted, the line numbers are re-numbered with the following bracketing values established for the record specification forms:

H starts at	10
F	6000
E	7000
L	8000
I	10000
C	30000
O	50000

If "BY inc" is specified, the source will be renumbered accordingly. Again, "inc" cannot be less than 1.

# RENUM

## *Parameter option 4: SO[URCE] W[ITH] SE[QUENCE]*

The RPG source line numbers are re-numbered to agree with the work file sequence numbers.

## *Parameter option 5: SE[QUENCE] W[ITH] SO[URCE]*

The work file sequence numbers are renumbered to agree with the RPG source line numbers. If each line number in the RPG source program is not in ascending order, the RENUM command will not renumber the work file.

## **Purpose**

The RENUM command re-numbers sequence line numbers in the work file and/or line numbers in the RPG source program (columns 1-5).

Execution mode: Line or Block

Control Y: Stops the listing of lines that are being renumbered; does not stop the renumbering.

## **EXAMPLES**

The following are legal abbreviations which may be used with the RENUMber command and its parameters:

R ALL	Renumber all sequence line numbers in the work file.
R BOTH BY 10	Renumber sequence line numbers and source line numbers using an increment value of 10.
R SO W SE	Renumber source line numbers so that they correspond with sequence line numbers.
REN SO	Renumber source line numbers so that they have bracketing values according to the record specification form with which they are associated.



The examples use a truncated version of SIMCAL consisting of the first five lines. It has been stored in a permanent file with the name PARTSIM. When an attempt to TEXT the file is made and the filename is typed incorrectly, an error message is returned (1). The command is then given correctly and the file contents are listed (2). RENUM is entered along with a parameter calling for an increment value of 100.

When the command has been executed, RISE returns a message (3). After the LIST command is executed, the renumbered lines are shown on the screen (4). RENUM BOTH follows, with the BY 1 parameter. Another listing reveals how both sequence line numbers and source line numbers are affected by this command (5).

```
(1) >TEX PARTSIM2
      NONEXISTENT PERMANENT FILE (FSERR 52)
(2) >TEX PARTSIM
      Text completed.
      >L ALL
          1      00011H
          SIMCAL
          2      00012FINPUT  IP  F      72          DISC
          3      00013FOUTPUT D  F      72          DISC
          4      00014IINPUT  AA  01    1 CA
          5      00015I      DR  02    1 CS
(3) >RENUM ALL BY 100
      Renum completed.
(4) >L ALL
      100      00011H
          SIMCAL
      200      00012FINPUT  IP  F      72          DISC
      300      00013FOUTPUT D  F      72          DISC
      400      00014IINPUT  AA  01    1 CA
      500      00015I      DR  02    1 CS
      >
      >RENUM BOTH BY 1
      Renum completed.
      >L ALL
          1      00001H
          SIMCAL
          2      00002FINPUT  IP  F      72          DISC
          3      00003FOUTPUT D  F      72          DISC
          4      00004IINPUT  AA  01    1 CA
(5) >L ALL
          5      00005I      DR  02    1 CS
      >
```

# RENUM

In the next example, a misspelling of SOURCE evokes an error (1). When the command is entered correctly and listed on the screen, the source line numbers are seen to be numbered according to the bracketing values (2). Next, another typing error causes the display of an error message (3). The command, entered correctly, causes the sequence line numbers to take on the same numbers as the source line numbers (4). The source line numbers are then changed by a command which sets their increment value at 10. The results are brought to the screen by a LIST command (5).

An error results from a mistaken parameter following WITH (6) and a message is displayed. When a command to renumber source lines with sequence line numbers is correctly entered, and the LIST command is used, the results are displayed on the screen (7).

```
(1) >RENUM SOUCE
      ^
Error 161: Syntax error on SOURCE.
>RENUM SOURCE
Renum completed.
>L FIR/L
  1      00010H
SIMCAL
  2      06000FINPUT  IP  F      72          DISC
  3      06001FOUTPUT D  F      72          DISC
  4      10000IINPUT  AA  01    1  CA
  5      10001I      DR   02    1  CS

(2) >
(3) >RENUM SEQNUM WITT SOURCE
      ^
Error 164: Syntax error on WITH.
>RENUM SEQNUM WITH SOURCE
Renum completed.
>L A
  10     00010H
SIMCAL
  6000   06000FINPUT  IP  F      72          DISC
  6001   06001FOUTPUT D  F      72          DISC
 10000   10000IINPUT  AA  01    1  CA
 10001   10001I      DR   02    1  CS
```

(example continued)

(4) >  
 >RENUM SOURCE BY .1  
 Error 153: Increment cannot be less than 1 for source.  
 >RENUM SOURCE BY 10  
 Renum completed.

(5) >L ALL  
 10 00010H  
 SIMCAL  
 6000 00020FINPUT IP F 72 DISC  
 6001 00030FOUTPUT O F 72 DISC  
 10000 00040IINPUT AA 01 1 CA  
 10001 00050I OR 02 1 CS

(6) >R SD W SD  
 Error 159: Expecting SEQNUM after WITH.  
 >R SD W SE  
 Renum completed.

>LIST A  
 10 00010H  
 SIMCAL  
 6000 06000FINPUT IP F 72 DISC  
 6001 06001FOUTPUT O F 72 DISC  
 10000 10000IINPUT AA 01 1 CA  
 (7) 10001 10001I OR 02 1 CS

# RUN

RUN executes a prepared program.

## Form

```
RU [N] progfile [,entrypoint] [;NOPRIV] [;LMAP] [;DEBUG]
                                [;MAXDATA=segsz]
                                [;PARM=num]
                                [;STACK=stacksize]
                                [;DL=dsize]
                                [;NOCB]
                                P
                                [;LIB= G]
                                S
```

The parameters, preceded by a semicolon, can appear in any order. The RUN command is similar to the MPE :RUN command. Because it is similar, the syntax of RISE's RUN command is exactly the same as MPE's RUN command syntax. Therefore, delimiters between parameters are required.

## Parameter explanation

progfile	Actual designator of program file that contains prepared program.
entrypoint	Program entry point where execution is to begin. May be primary entry point of program or any secondary entry point in program's outer block. Default is primary entry point.
NOPRIV	Declaration that program segments will be placed in non-privileged mode capability. This parameter is intended for programs prepared with privileged mode capabilities. Normally, programs containing privileged instructions are executed in privileged mode only if the program was prepared with privileged mode. If NOPRIV is specified in the :RUN command, all program segments are placed in non-privileged mode. Library segments are not affected because their mode is determined independently. (Note that a program containing legally compiled privileged code, placed in a non-privileged mode, may abort when you attempt to execute it.)

- LMAP Request to produce a descriptive listing of the allocated (loaded) program on file whose formal designator is LOADLIST. If no :FILE command is found that references LOADLIST, listing is sent to \$STDLIST. Default is no listing.
- DEBUG Request to issue a Debug call before the first executable instruction of the program. This parameter is ignored when a non-privileged user runs a program having privileged mode capability. This parameter is also ignored if the user does not have read and write access to the program file. Default is that DEBUG call is not issued.
- MAXDATA= Maximum stack area (Z-DL) size permitted, in words. This parameter is included if the size of DL-DB or Z-DB areas will be changed during the program execution. Default is MPE assumes areas will not be changed.
- PARM= Value that can be passed to program as a general parameter for control or other purposes. When program is executed, this value can be retrieved from address Q (initial) -4 where Q (initial) is Q address for outer block of program. Value can be octal number or signed or unsigned decimal number. Default is Q (initial) -4 address is filled with zeros.
- stacksize Size of initial local data area (Z-Q (initial) ) in stack. This value must exceed 511 words, and override stacksize estimated by MPE Segmenter. Default is estimated by Segmenter.
- DL= DL-DB area to be initially assigned to stack. This area is of interest mainly in programmatic applications. In all cases, the DL-DB area is rounded upward so that the distance from the beginning of the stack data segment to the DB address is a multiple of 128 words. Default is estimated by the Segmenter.
- NOCB Request that file system not use stack segment (PCBX) for its control blocks, even if sufficient space is available. This permits expansion of the stack (with the DLSIZE and ZSIZE intrinsics) to the maximum possible limit at a later time, but causes the file management system to operate more slowly for this program.



# RUN

- LIB=G Search segmented procedure libraries to satisfy external references during allocation in the following order: group library, account public library, system library. Default is S.
- LIB=P Search segmented procedure library to satisfy external references during allocation in the following order: account public library, system library. Default is S.
- LIB=S Search system library only to satisfy external references during allocation. This is the default.

## Purpose

The RUN command allows you to execute a prepared program. It operates in the same way that the MPE command :RUN operates. Note that you can use the VERIFY RPG and VERIFY PREP commands to compile and prepare your work file into a program file to be executed with this RUN command. (See VERIFY.)

Execution mode: Line

## EXAMPLES

The following are legal abbreviations which may be used with the RUN command and its parameters:

- |                          |  |
|--------------------------|--|
| RU FCOPY.PUB.SYS         | Run the FCOPY utility.   |
| RU \$OLDPASS;LIB=G;DEBUG | Run the prepared program \$OLDPASS; search segmented procedure libraries beginning with the group library to satisfy external references during allocation; issue a DEBUG call before the first executable instruction of the program. |

In the example the RUN command is used to run two prepared programs. SIMCAL is to be used as the simple calculator it was designed to be, and it must be compiled, prepared, and run. Assume that SIMCAL was compiled into a USL file named \$OLDPASS with RISE's VERIFY RPG command. Now it must be prepared using the Segmenter (or, alternatively, with the VERIFY PREP command). Initially, the command to run the Segmenter subsystem is given (1). The HP banner is then brought to the screen (2). The USL file is identified (3), and then prepared in a new program file (4). The user then leaves the Segmenter subsystem with the EXIT command and RISE's banner is displayed (5).

Next, the input and output file equations are given for SIMCAL. Note that the user must supply the colon (: ) prompts (6). Following that, the RUN \$OLDPASS executes SIMCAL (7).

Examples of SIMCAL's application follows: it is used to add (8), subtract (9), multiply (10), and divide (11). The colon entered when the calculations are finished returns the user to RISE (12).

```
(1) >RUN SEGQVR.PUB.SYS
(2) HP32050A.01.03 SEGMENTER/3000 (C) HEWLETT-PACKARD CO. 1979
(3) -USL $OLDPASS
(4) -PREP $NEWPASS
(5) -EXIT

      HP32104A.05.00 RPG INTERACTIVE SYSTEM ENVIRONMENT   RISE/3000
      (C) HEWLETT-PACKARD CO. 1981           WED, SEP 16, 1981, 10:29 AM

(6) >:FILE INPUT=$STDIN
    >:FILE OUTPUT=$STDLIST
(7) >RUN $OLDPASS

      SIMPLE                CALCULATOR

      ENTER DATA IN THE FORMAT:
      C NNNDD NNNDD
(8)  A 11100 11100

      ..... YOUR PROBLEM:  111.00 ADD  111.00 EQUALS      222.0000
```

# RUN

ENTER DATA IN THE FORMAT:

C NNNDD NNNDD

(9) S 22222 11111

..... YOUR PROBLEM: 222.22 SUB 111.11 EQUALS 111.1100

ENTER DATA IN THE FORMAT:

C NNNDD NNNDD

(10) M 01000 01000

..... YOUR PROBLEM: 10.00 MULT 10.00 EQUALS 100.0000

ENTER DATA IN THE FORMAT:

C NNNDD NNNDD

(11) D 10010 01000

..... YOUR PROBLEM: 100.10 DIV 10.00 EQUALS 10.0100

ENTER DATA IN THE FORMAT:

C NNNDD NNNDD

(12) :

HP32104A.05.00 RPG INTERACTIVE SYSTEM ENVIRONMENT RISE/3000  
(C) HEWLETT-PACKARD CO. 1981 WED, SEP 16, 1981, 10:29 AM

>



# SHOW

SHOW displays a page of your source file for direct editing on the screen.

<b>Form</b>
-------------

S [HOW] [lb] [C [OL]]
-----------------------

## Parameter explanation

- lb            First line of the page you wish to have displayed. (If omitted, the first page of the work file is displayed.)
- COL           Display column indicator rather than RPG Record Specification form. (15 lines will be displayed with the COL parameter.)

## Purpose

The SHOW command displays a page (10 lines) for direct screen editing in Block Mode. If you do not specify a line number ("lb"), the first page of your work file will be displayed. If "COL" is specified, a page of 15 lines is displayed with the column indicator. Otherwise, the page appears with the RPG Record Specification form appropriate to the line or lines being displayed. (An exception occurs when column 6 of a line contains an illegal RPG form entry. The screen display defaults to a column indicator in such an instance.)

In SHOW mode the terminal's cursor control keys and two of the editing keys —INSERT CHAR [acter] and DELETE CHAR [acter] —will prove particularly useful in your editing. (INSERT LINE and DELETE LINE keys are not activated.) Also, tabs have been set to correspond with the important Record Specification form columns. Using the TAB key rather than the space bar to position your cursor, you move quickly from column to column to the position where changes are necessary. You then type your changes in directly.

When you are satisfied with your changes, press the ENTER key to update the work file with your newly edited page.

# SHOW

## Related commands

Two sets of special function key labels can appear at the top of the screen with the SHOW command. They are the Master set and the Scroll set. To change sets, you press softkey F1, which is a toggle switch.

When the Master set is activated, the special function keys perform the following functions:

- F1     switch to Scroll set
- F2     insert lines
- F3     show page with or without edit sequence numbers
- F4     move cursor to Command Window
- F5     scroll forward to next page
- F6     scroll backward one page
- F7     enter Line Mode
- F8     display main menu

Softkey F3 will display the RPG source file on the screen with or without the sequence line numbers of the work file. If F3 is on, the sequence numbers will appear on the left hand side of the screen.

RPG source records are 80 columns long. When RISE displays the source records on the screen without the sequence numbers, the individual records will fit on a single line. When you press F3, calling for a display of sequence numbers, columns 72-80 of each record are not shown. These columns are not deleted in such an instance, but the screen no longer contains the complete record since room is made to display the sequence numbers on the lefthand side.

Thus, you may display complete source records without the sequence numbers when you wish. If, however, you wish to determine where you are in the work file, and need sequence numbers to accomplish this, you will do so at the cost of truncating columns 72-80 from the screen display. They are not accessible at this time and cannot be altered.

### **Inserting Lines**

When you wish to insert new lines between existing lines you may have to make room for them. If this is necessary you press F2 when the Master set of function keys is activated. This changes the special function key labels so that F1 through F4 are prepared to receive insertion commands. You then position the cursor at the line preceding the position where you wish to begin the insertion.

When the cursor is positioned correctly, you press softkeys F1, F2, F3, or F4, depending on how many lines you are adding (F1 adds 1 line; F4 adds 4 lines). An example of the correct procedure for inserting lines is included in the examples below.

Note: If the terminal is unexpectedly locked out at this point, see the section on Recovery Procedures.

### **Scroll Set**

While in SHOW mode, you can increase your scrolling capabilities by switching to Scroll set. Again, this is accomplished by pressing F1.

# SHOW

The softkeys have the following functions when Scroll set is activated:

- F1     switch to Master set
- F2     display first page
- F3     display last page
- F4     move cursor to Command Window
- F5     scroll forwards a page
- F6     scroll backwards a page
- F7     scroll forwards 1/2 page
- F8     scroll backwards 1/2 page

Execution mode: Block

## EXAMPLES

The following are legal abbreviations which may be used with the SHOW command and its parameters:

- S F C        Enter Show mode displaying the first line of the source file with the column indicator.
- S 10        Enter Show mode and display line 10 with the appropriate Record Specification form.
- SHO        Enter Show mode.

In the first example, the SHOW FIRST command brings the Header Specification form to the screen along with the first line of the source file (1). In the second display, the cursor has been moved into position and new data is added to the first line. Note that RISE is now in Block mode and the ENTER key must be pressed to make changes to the first line permanent (2).

> SHOW FIRST

(1)

STEP SCROLL	INSERT LINES	REDNUM OFF	FORWARD WINDOW	SCROLL FORWARD	SCROLL BACKWARD	LINE MODE	MAIN MENU
----------------	-----------------	---------------	-------------------	-------------------	--------------------	--------------	--------------

RPG HEADER SPECIFICATION				FILE TRANS X REF	COPTPL TYPE
FORM DATE FILE NO INVT	YIN LOKUP	IND SET	FLAGE	SEQ CHECK	
TYPE DEBUG OPT	SEQ 1ST	ALT COL	F POS	PASS	ERROR LGS
LINE DUMP	JREC	S.O.E	SIGN S	DOLSEP	ERROR NUMBERS
NUM HFILENAME1 F ID			SNOB B	SUBSEP	LSN 3 5 7 9 1 3 5 7
00011H			1TF	S	1X1NS-V-V-V-V-V-V-V-V
					PROG NAME SIMPAL

COMMAND: \_\_\_\_\_

ENTER

SHOW MODE



In the next examples, the special function key labeled SCROLL FORWARD is used while the Header Specification form is displayed. The first time the scrolling occurs, the RPG File Specification form appears on the screen with the associated source file lines (1). The next form to be brought to the screen by the SCROLL FORWARD softkey is the RPG Input Specification form. Again, the associated lines are also displayed (2).

(1)

SYSTEM	USER	PRINT	SEARCH	EDIT	SCREEN	LINE	MAIN
SCROLL	INES	OFF	WINDOW	FORWARD	BACKWARD	MODE	MENU

RPG FILE SPECIFICATIONS											
DESIGNATION	FILE	SEQ	LEN	FILE	ORG	OVER	EXT	CODE	CONT	LABEL	OPTION
FILE TYPE	IS	SEQ	LEN	FILE	ORG	OVER	EXT	CODE	CONT	LABEL	OPTION
FILE NAME	OP	REC	PPDC	A	FLOW	KEY	EDEVICE	SE	3-9	OPTION	TARGET
FORM	UC	FMT	MODE	P	KEY	EDEVICE	SE	3-9	OPTION	TARGET	1- USWITCH
LINE	DT	AFBLY	REC	1-K	KEY	EDEVICE	SE	3-9	OPTION	TARGET	1- USWITCH
NUM F	CD	EDVLEN	LEN	R99I	LDC	LCLASS	TRANS				32
00012	INPUT	I	F	73			DISC				
00013	OUTPUT	O	F	73			DISC				

COMMAND: \_\_\_\_\_

ENTER \_\_\_\_\_

MAIN MODE \_\_\_\_\_

# SHOW

(2)

MASTER+ SCROLL    INSERT LINES    SEQNUM \* OFF \*    COMMAND WINDOW    SCROLL FORWARD    SCROLL BACKWRD    LINE MODE    MAIN MENU

RPG INPUT SPECIFICATIONS									
REC INCL	LA	REL	CHAR	CHAR	CHAR	STACK SEL	MATCHING	FIELD	RECORD REL
OPTION			PORT	PORT	PORT				
NUM OF RECS			NOT	NOT	NOT	DATA ENT	CHAINING	PLUS IND	
FORM	GROUP	SEQ	POS	C	POS	C	POS	C	DEC CONTROL
LINE	FILE	OR	1 - Z	1 - Z	1 - Z	1 - Z	FROM	TO	0 FIELD L1
NUM	INAME	AND NO	9999ND	9999ND	9999ND	2			9 NAME L9
									MINUS IND
									DEPO ELEM
00014I	INPUT	AA	01	1	CA				
00015I		OR	02	1	CS				
00016I		OR	03	1	CM				
00017I		OR	04	1	CD				
00018I		OR	05						
00019I						3	730PRND1		
00020I						3	1320PRND2	33	



COMMAND: [REDACTED] ]  
ENTER [REDACTED]  
SHOW MODE [REDACTED]





# SHOW

In the next example, the screen is switched to Scroll Set following the insertion by pressing special function key #1 (MASTER SCROLL). Note that the asterisk in the softkey label for F1 is now next to "SCROLL". This indicates that the Scroll Set of keys are now active (1). Special function key #3 (SCROLL LAST) produces the next screen display, which contains the final lines of the source file associated with the RPG Output Specifications form (2).

(1)

MASTER SCROLL*	SCROLL FIRST	SCROLL LAST	SCROLL RANDOM	SCROLL FORWARD	SCROLL BACKWARD	SCROLL /2 FWD	SCROLL /2 BWD
----------------	--------------	-------------	---------------	----------------	-----------------	---------------	---------------

```

RPG INPUT SPECIFICATIONS
-----
OPTION          PORT          PORT          PORT          STACK BBS  MATCHING/  FIELDS RECORD FEL
NUM OF RECS    NOT          NOT          NOT          DATA FMT  CHAINING  PLUS IND
GROUP BBS     POS C      POS C      POS C      FROM TO  OFIELD L1  MINUS IND
LINE FILE OR 1 1 -      1 -      1 -      9NAME L9  ZERD/BLANK
NUM INAME AND 9999ND 9999ND 9999ND 2
000141 INPUT  AA  01  1  0A
000161      DP  03  1  0E
000181      DP  03  1  00
000171      DP  14  1  02
000191      DP  0E
.
.
.
000181      000181
000201      000201
-----
COMMAND: [REDACTED]
ENTER
*END MODE
  
```

(2)

SPACE AFTER	SPACE BEFORE	SS/FETCH	TYPE	ADD	DET	IND	EDIT	BY/AN	END	NO CR	COMMENT
EF	OUT	IND	EDIT	BY/AN	END	NO CR	COMMENT				
W	R	T	T	T	T	T	T	T	T	T	T

```

SPACE AFTER SPACE BEFORE SS/FETCH TYPE ADD DET IND EDIT BY/AN END NO CR COMMENT
W R T T T T T T T T T T T
000380
000400 DP
000410
000420
000430
000440 DP
000450
000460
000470
000480
000490
000480*END OF PROGRAM
-----
COMMAND: [REDACTED]
ENTER
*END MODE
  
```



# SHOW

To produce the first display in the final example, the user has entered SHOW FIRST COL command or some valid abbreviation. Also, special function key #3 has been pressed while in Master Set (this key toggles sequence numbers on and off). A fifteen line "page" is brought to the screen and with it the sequence line numbers which RISE keeps for use in referencing records. Note the command in the command window (1). The next display follows the execution of the command, SHOW 11, after the user hits the ENTER key. The column indicator form is cleared and the RPG Calculation Specifications form, which is appropriate to line 11, is brought to the screen. Sequence line numbers continue to be displayed and will be until special function key #3 is pressed (2).

(1) MASTER\*    INSERT    SEQNUM    COMMAND    SCROLL    SCROLL    LINE    MAIN  
SCROLL    LINES    \* ON \*    WINDOW    FORWARD    BACKWRD    MODE    MENU

SEQ	1	2	3	4	5	6	7
NUM ...	12345678901	2345678901	2345678901	2345678901	2345678901	2345678901	2345678901
00001000	00011H						
00002000	00012F	INPUT	IP	F	72	DISC	
00003000	00013F	OUTPUT	O	F	72	DISC	
00004000	00014I	INPUT	AA	01	1	CA	
00005000	00015I		DR	02	1	DS	
00006000	00016I		DR	03	1	DM	
00007000	00017I		DR	04	1	DD	
00008000	00018I		DR	05			
00009000	00019I					3 72DPRND1	
00010000	00020I					9 132DPRND2	99
00011000	00021C	01	DPRND1	ADD	DPRND2	RESULT	104
00012000	00022C	02	DPRND1	SUB	DPRND2	RESULT	
00013000	00023C	03	DPRND1	MULT	DPRND2	RESULT	
00014000	00024C	04N99	DPRND1	DIV	DPRND2	RESULT	H
00015000	00025C	04 99		Z-ADD0		RESULT	

COMMAND: [SHOW 11]  
ENTER?  
SHOW MODE



# TEXT

TEXT copies a permanent file into a work file for editing.

## Form

```
T [EXT] filename [N [OW] ]
```

## Parameter explanation

filename      Name of the permanent file you are copying to the work file.

N [OW]      Your work file contains the contents of one file but you may wish to TEXT another without saving the contents of that work file. The contents of the file you wish to TEXT will not be copied unless you KEEP the current work file to a permanent file. You can override the need to KEEP the file by using the NOW parameter. In doing so you discard your changes. If you are directly editing a file (using the FILE command) and you give the TEXT command, the file will be saved (closed) first automatically, whether you specify "NOW" or not.

If you choose not to use the "NOW" parameter, RISE displays a safety message asking you if it is all right to clear the current work file. You may respond with "Y [ES]" to clear the current file and text in another. If you respond with "N [O]", the TEXT command is cancelled so that you may perform a KEEP first if you wish.

## Purpose

The TEXT command copies the contents of a permanent file into a work file for additions and changes. You make additions and changes to a work file only, which you save after editing with the KEEP command.

### Related commands

See the discussion of the KEEP command. If the file is unnumbered, RISE automatically appends the sequence numbers in columns 81-88. If the text file is already numbered, RISE uses the existing sequence numbers. Text files are compatible with EDIT/3000 files.

Execution mode: Line or Block

Record pointer: At first line in work file

## EXAMPLES

The following are legal abbreviations which may be used with the TEXT command and its parameters:

- |           |  |
|-----------|--|
| T FILE1 N | Copy the contents of a permanent file named FILE1 into the work file immediately, without displaying safety prompts or saving changes made to the work file.                           |
| TEX FILE1 | Copy the contents of the permanent file named FILE1 into the work file, displaying safety prompts if the current work file has been edited and has not be saved with the KEEP command. |

# TEXT

In the first example, SIMCAL is copied into the work file using the TEXT command, and an execution message is returned (1). The first five lines of the file are listed, and then lines 1 through 3 are deleted (2). An attempt to TEXT SIMCAL with these changes receives a safety prompt (3). When it is answered negatively, the command is cancelled (4). Next, the KEEP command is used with the filename BACKUP. Thus, BACKUP without the three lines, is now stored as a permanent file (5).

```
(1) >TEXT SIMCAL
    Text completed.
    >L 1/5
      1      00011H
      SIMCAL
      2      00013FINPUT  IP  F      72      DISC
      3      00013FOUTPUT 0  F      72      DISC
      4      00014IINPUT  AA  01    1  0A
      5      00015I      DR  02    1  05

(2) >D 1/3
      1      00011H
      SIMCAL
      2      00013FINPUT  IP  F      72      DISC
      3      00013FOUTPUT 0  F      72      DISC
    Delete completed.

(3) >T SIMCAL
    KEEP not done, CLEAR current work file?NO
    Text cancelled.

(4) >KEEP BACKUP
    keep completed.

(5) File name is BACKUP
```



In the second example, the CHANGE command is used to replace one four-character string with another. The results of the command, and an execution message, are then displayed (1). The TEXT command is used with the NOW parameter and is executed (2). Note that no safety prompts were issued even though changes were previously made. When LIST 1/8 is given, the first three lines are missing from the listed lines because they were deleted in the first example (3). Next, line 5 is modified (4). The TEXT command used with SIMCAL evokes the safety prompts because of the changes made to the file. When the prompt is answered in the affirmative, the command is executed (5).

```

>CHANGE :DISC: TO :TAPE: IN ALL
      00013RINPUT  2P  F      72
      00013ROUTPUT 0  F      73
(1)  Changes completed.
      >TEXT BACKUP NOW
(2)  Text completed.
      >
(3)  >L1/8
      4      00014IINPUT  AA  01  1 CA
      5      00015I      DR  02  1 CS
      6      00016I      DR  03  1 CM
      7      00017I      DR  04  1 CD
      8      00018I      DR  05
(4)  >M5
      5      00015I      DR  02  1 CS
           :      RAND
      5      00015I      AND  02  1 CS
           :
(5)  >TEXT SIMCAL
      KEEP not done, CLEAR current work file?YES
      Text completed.
      >

```

# UNDEL

UNDELeTe restores lines that were erroneously deleted.

Form U [NDEL]
------------------

## Purpose

The UNDELeTe command restores the lines which were removed by the last DELETE command. It does not restore characters which were deleted while in the MODIFY mode.

If you DELETE lines, then ADD new ones with the same numbers as those deleted, the deletions will not be restored by the UNDEL command. It only restores lines when the numbers of the lines to be restored do not already exist in the work file. For example, if you DELETE lines 1/5, ADD 1/3, then try to recover with the UNDEL command, RISE will not be able to restore lines 1/3 because the lines exist due to your new additions. In this case, no lines will be restored unless you move 1/3 to another location, freeing those locations for full restoration.

## Related commands

In SHOW Mode, changes are not permanent until the ENTER key is depressed, at which time RISE changes the internal page buffer and updates the work file with it. If, previous to pressing the ENTER key, you made changes which you wish to cancel, you do not use the UNDEL command. You scroll forward or backward to a different page, then scroll back to the page on which the undesired changes were made. RISE will restore that page to its original state from the internal buffer which never changed.

Execution mode: Line or Block

Record pointer: At last line undeleted

Control Y: Stops listing the lines that are being undeleted.

## EXAMPLES

The following are legal abbreviations which may be used with the UNDELete command:

U           Undelete lines removed by the last delete command.  
UND

In the first example, lines 1 through 5 of SIMCAL are listed and then deleted. An execution message appears at the terminal when the deletion is completed (1). The UNDEL command is then used, and the lines previously deleted are restored as they were (2). Lines 5 through 10 are next deleted (3).

```

>L1/5
 1      00011H
SIMCAL
 2      00012FINPUT  IP  F      72          DISC
 3      00013FOUTPUT D   F      72          DISC
 4      00014IINPUT  AA  01     1 CA
 5      00015I      DR   02     1 CS
>D1/5
 1      00011H
SIMCAL
 2      00012FINPUT  IP  F      72          DISC
 3      00013FOUTPUT D   F      72          DISC
 4      00014IINPUT  AA  01     1 CA
 5      00015I      DR   02     1 CS
(1) Delete completed.
>
>UNDEL
 1      00011H
SIMCAL
 2      00012FINPUT  IP  F      72          DISC
 3      00013FOUTPUT D   F      72          DISC
 4      00014IINPUT  AA  01     1 CA
 5      00015I      DR   02     1 CS
(2) Undel completed.
>
>D 5/10
 5      00015I      DR   02     1 CS
 6      00016I      DR   03     1 CM
 7      00017I      DR   04     1 CD
 8      00018I      DR   05
 9      00019I          3  720PRND1
10     00020I          9  1320PRND2

```

(3) Delete completed.

# UNDEL

In the next example, a file named RPGDOC — used previously in the JOIN example — is JOINed to the work file beginning at line 5 (1). The UNDEL command is then tried and rejected by RISE. This occurs because line 5, which was deleted in the first example, has been used again through the JOIN command. A message appears explaining why the UNDELETE command can not be executed (2). The lines containing RPGDOC are then MOVEd to a new position, beginning at line 10.5 (3). When this is done, line 5 no longer exists in the file. Now the UNDELETE command, which was previously rejected, is executed by RISE (4).

```
>JOIN RPGDOC TO 5
  5          H*
  5.1       H* PROGRAM NAME:
  5.2       H*
  5.3       H* PROGRAMMER  :
  5.4       H* DATE WRITTEN:
  5.5       H* PURPOSE    :
  5.6       H*

(1) Join completed.
(2) >UNDEL
      5      00015I      DR  02  1 CS
      Can't Un-delete a line, the line number exists already.
>
(3) >MOVE 5/5.6 TO 10.5
      10.5     H*
      10.51    H* PROGRAM NAME:
      10.52    H*
      10.53    H* PROGRAMMER  :
      10.54    H* DATE WRITTEN:
      10.55    H* PURPOSE    :
      10.56    H*
      Move completed.
(4) >U
      5      00015I      DR  02  1 CS
      6      00016I      DR  03  1 CM
      7      00017I      DR  04  1 CD
      8      00018I      DR   05
      9      00019I
      10     00020I
                                     3  720PRND1
                                     9  1320PRND2
                                     99

      Undel completed.
>
```

# ● VERIFY

VERIFY compiles a work file, lists a compilation, or prepares a USL file into a program file.

```
Form
V [ERIFY] R [PG]
           L [IST]
           P [REP] [profilename] [;ZERODB] [;PMAP]
                 [;MAXDATA=segsz] [;STACK=stacksz]
                 [;DL=dlsz] [;CAP=caplist] [;RL=filename]
                 [;PATCH=patchsz]
```



## ● Parameter explanation

- R [PG]      Calls the RPG compiler to compile the work file into the USL file named "\$OLDPASS", a system-defined file name. Afterwards, it causes the newly compiled listing to be displayed on the screen. Using special function keys, you can scroll through the file in split screen. The USL file in \$OLDPASS is not created if any serious compilation errors occur.
- L [IST]     Displays the last compilation listing completed as a result of the V [ERIFY] R [PG] command.
- P [REP]     Calls the Segmenter to prepare a USL file in \$OLDPASS to the program file named "progfile". If "progfile" is omitted, the program file will default to \$OLDPASS. (During the preparation of the USL file, the Segmenter uses \$NEWPASS as the program file name. Afterwards, \$NEWPASS is renamed \$OLDPASS.)
- progfile    The name of the file onto which the prepared segments are to be written. If the file named does not exist, the Segmenter will build a job temporary file for you. (Note: code segments in a program file cannot lie across disc extent boundaries. Thus, all segments in such files must be constructed within one extent. See the MPE Intrinsic Reference Manual for a discussion of disc extents.)
- ZERODB     An indication that the initially-defined DL-DB area, and uninitialized portions of DB-Q (initial) area will be initialized to zero. If this parameter is omitted, these areas are not affected.

## VERIFY

- PMAP An indication that a listing describing the prepared program will be produced with the formal filename of "SEGLIST". If there is no file equation for "SEGLIST", the listing is printed on your terminal. If the parameter is omitted, no listing is produced.
- MAXDATA= Maximum stack area (Z-DL) size permitted, in words. This parameter is included if you expect to change the size of the DL-DB or DB-Z areas during process execution. If omitted, MPE assumes that these areas will not be changed.
- segsize
- STACK= The size of the user's initial local data area (Q (initial) to Z) in the stack, in words. This overrides the stacksize estimated by the Segmenter, which applies if the stacksize parameter is omitted. (The default is a function of estimated stack requirements for each program unit in the program.) Since it is difficult for the system to predict the behavior of the stack at run time, you may want to override the default by supplying your own estimate with stacksize.
- stacksize
- DL= The DL-DB area to be initially assigned to the stack. If the "dlsiz" parameter is omitted, a value of zero is used.
- dlsiz
- CAP= The capability-class attributes associated with the caplist user's program; specified as two-character mnemonics. If more than one mnemonic is specified, each must be separated from its neighbor by a comma.
- caplist

The mnemonics are:

- IA = Interactive access
- BA = Local batch access
- PH = Process handling
- DS = Data segment management
- MR = Multiple resource management
- PM = Privileged-mode operation

Users who issue the PREPare command can only specify capabilities that they themselves possess (through assignment by the Account Manager). If the user does not specify any capabilities, only IA and BA (if the user possesses them) will be assigned to this program.

RL = filename      The name of the Relocatable Library (RL) to be searched to satisfy external references during preparation. This can be any permanent file of type RL. It need not belong to the log-on group, nor does it have a reserved, local name. This file yields a single segment that is incorporated into the segments of the program file prepared. If "filename" is omitted, no library will be searched.

PATCH = patchsize      The number of extra words added to each code segment before the Segment Transfer Table to allow room for code patches to a segment.

For the stacksize, dsize, and maxdata parameters, a value of -1 denotes the default (equivalent to omitting the parameter).

Note that the optional parameters associated with the P [REP] command can be used in any order. When the Segmenter is executing, its HP banner and all its responses will appear on your terminal. Moreover, because RISE's PREP option is exactly the same as the Segmenter's PREPARE command, the syntax is the same and delimiters are necessary between parameters.

**Purpose**

The VERIFY command has three functions. It calls the RPG compiler to compile the work file and, when the compilation is finished, displays the newly compiled file listing. Also, it lists the last compilation listing produced by the VERIFY command and its RPG parameter. Finally, it calls the Segmenter to prepare a USL file in \$OLDPASS to a program file. Therefore, you may compile, prep, and execute your RPG source program in the work file by using the VERIFY RPG, VERIFY PREP, and RUN commands. You may also use the :MPEcommand to set up any file equations necessary to execute your program correctly. For instance:

```
:FILE OUTFILE=$STDLIST
```

# VERIFY

## Related commands

The special function keys are redefined by the VERIFY RPG and LIST commands to facilitate management of the compilation listing. The keys function as follows:

- F1     switch window to split screen — Top or Bottom— or Full Window
- F2     display first page of compilation listing
- F3     display last page
- F4     find error in compilation listing
- F5     scroll forwards one page
- F6     scroll backwards one page
- F7     scroll forwards 1/2 page
- F8     scroll backwards 1/2 page

Softkey F1 allows you to divide the screen up into two fixed portions—a top and bottom half—or join them in a full page display. Pressing F1 will switch the screen display in the following order: Top Window, Bottom Window, and back to Full Window. The F1 label indicates what will be displayed next on the screen if you press F1. In other words, when Top Window is activated, the label will read Bottom Window. If you press F1, RISE will respond by activating the Bottom Window.

Futhermore, a dividing line separating the two halves of the screen indicates the activated half. Up arrows (^) point toward the Top Window; a line of v's indicates the Bottom Window.

## Use of Split Screen

The split screen allows you to view, simultaneously, two parts of the compilation listing at one time. For example, one window can show error messages while the other shows the corresponding source line containing the error.

Special function key F4, the “Find Error” key, is extremely useful when you use it with Top Window mode. After you press F4, RISE will display the next RPG source line with errors or warnings in the Top Window while automatically displaying the corresponding error message at the bottom.

Softkey F4 finds and displays the next source line containing an RPG error or warning. The search for the error begins after the last line displayed on the current window. F4 only finds errors for which the RPG compiler emits an error number, and only on the source line containing that error.



PRINT: the PRINT command with the RPGLIST parameter will print the compilation listing offline.

```

                RPG
Execution mode: VERIFY      Block mode
                LIST
                VERIFY PREP  Line mode
    
```

## EXAMPLES

The following are legal abbreviations and can be used with the VERIFY command and its parameters:

- V R            The RPG compiler will compile the edit file.
- V L            The last compilation listing produced by the V [ERIFY] R [PG] command will be displayed on the screen.
- V PREP        The segmenter will prepare a USL file in \$OLDPASS into a program file.

In the first example, five lines of SIMCAL are listed on the screen. The work file containing SIMCAL is then compiled when the VERIFY RPG command is given and the results are displayed (1). Note that the Command Mode window contains a message concerning serious errors and warnings after the VERIFY RPG command is executed (2).

A similar five-line listing follows (3). However, errors have been imbedded in these lines for the sake of the example. When these lines are compiled, the screen again exhibits the compilation, but the message in the Command Mode window reports on the errors and warnings (4).

```

>
>L1/5
 1      00011H
SIMCAL
 2      00012F INPUT   IP   F                DISC
 3      00013F OUTPUT   D                DISC
 4      00014I INPUT   AA   01   1  XA
 5      00015I         DR   02   1  CS
(1) >VERIFY RPG
    
```

# VERIFY

TOP WINDOW    SCROLL FIRST    SCROLL LAST    FIND ERROR    SCROLL FORWARD    SCROLL BACKWRD    SCROLL 1/2 FRD    SCROLL 1/2 BKD

PAGE 0001    HEWLETT PACKARD 32104A.04.07    PPG/3000 (C) HEWLETT-PAC  
KARD CO. 1975. THU. MAY 28. 1981.    3:08 PM

```
0001    00011H  
          SIMCAL  
0002    00012FINPUT    IP    F            72            DISC  
0003    00013FOUTPUT    O    F            72            DISC  
  
0004    00014IINPUT    AA    01    1    CA  
0005    00015I            DR    02    1    CS  
0006    00016I            DR    03    1    CM
```

COMMAND: [ ]  
ENTER?

(2) COMMAND MODE    SERIOUS ERRORS 000    WARNINGS 000

(3) >L1/5

```
1        00011H  
      SIMCAL  
2        00012FINPUT    IP    F            72            DISC  
3        00013FOUTPUT    O    F            72            DISC  
4        00014IINPUT    AA    01    1    XA  
5        00015I            DR    02    1    CS
```

>VERIFY RPG

(4) TOP WINDOW    SCROLL FIRST    SCROLL LAST    FIND ERROR    SCROLL FORWARD    SCROLL BACKWRD    SCROLL 1/2 FRD    SCROLL 1/2 BKD

PAGE 0001    HEWLETT PACKARD 32104A.04.07    PPG/3000 (C) HEWLETT-PAC  
KARD CO. 1975. THU. MAY 28. 1981.    3:12 PM

```
0001    00011H  
          SIMCAL  
0002    00012FINPUT    IP    F            72            DISC  
                          208W  
0003    00013FOUTPUT    O    F            72            DISC  
                          208W  
  
0004    00014IINPUT    AA    01    1    XA
```

COMMAND: [ ]  
ENTER?

COMMAND MODE    SERIOUS ERRORS 001    WARNINGS 005

The next example follows the previous one directly. Special function key #1 (TOP WINDOW) has been pressed and the screen is split (1). In the bottom half of the screen, error and warning message numbers appear beneath the lines containing the errors (2). Now, if special function key #4 (FIND ERR+MSG) is pressed, the usefulness of the split screen is demonstrated. In the second display, the top portion of the screen contains the compiled source lines and the error and warning numbers (3). The bottom window contains the numbers and their messages (4). Note that the up arrows (^) indicate that the screen is presently in top window mode.

In the next display, the same information is shown on the screen, but it is now in bottom window mode, as the down arrows (v) indicate. Pressing F1 caused the shift to bottom window mode (5).

```

(1) BOTTOM WINDOW   SCROLL FIRST   SCROLL LAST   FIND ERR+MSG   SCROLL FORWARD   SCROLL BACKWRD   SCROLL 1/2 FRD   SCROLL 1/2 BKD

PAGE 0001   HEWLETT PACIARD 33104A.04.07   PPS 0000 (0) HEWLETT-PAC
IARD CO. 1978. THU. MAY 28. 1981. 3:12 PM

0001   0001H
          SIMCAL

*****
0002   00013FINPUT   IF   F           0180
          008W
0003   00013FOUTPUT  D   78           0180
          008W

0004   00014FINPUT   AA   01           0180

COMMAND: [ ]
ENTER

COMMAND MODE   SERIOUS ERRORS 001   WARNINGS 005

```

(example continued)



In the next example, it is assumed that no serious compilation error occurred so the user proceeds to prepare and execute. The VERIFY PREP option is employed and the Segmenter banner is displayed (1). Next, the MPE command :SAVE assigns a new file designator to \$OLDPASS when it is made permanent (2). A second MPE command, :LISTF, with the 2 parameter, brings to the screen file information including file name, file code, record size, record format, data format (3). The input/output file equations are given (4), and the RUN command is issued using the new file designator (5). Again, the calculator is applied (6) and a colon, entered by the user (7), returns the user to RISE (8).

(1) >VERIFY PREP  
 HP32050A.01.03 SEGMENTER/3000 (C) HEWLETT-PACKARD CO. 1979  
 ---

Segmenter completed.

(2) >:SAVE \$OLDPASS,SIMPRDG  
 >:LISTF SIMPRDG,2  
 ACCOUNT= SUBSYS GROUP= RPG

	FILENAME	CODE	-----LOGICAL RECORD-----			-----SPACE-----		
			SIZE	TYP	EOF	LIMIT R/B	SECTORS #X MX	
(3)	SIMPRDG	PRDG	128W	FB	12	12	1	13 1 1

(4) >:FILE INPUT=\$STDIN  
 >:FILE OUTPUT=\$STDLIST  
 (5) >RUN SIMPRDG

SIMPLE CALCULATOR

(6) ENTER DATA IN THE FORMAT:  
 C NNNDD NNNDD  
 A 00111 00222  
 ..... YOUR PROBLEM: 1.11 ADD 2.22 EQUALS 3.3300

(7) ENTER DATA IN THE FORMAT:  
 C NNNDD NNNDD  
 :

(8) HP#####X.00.00 RPG INTERACTIVE SYSTEM ENVIRONMENT RISE/3000  
 (C) HEWLETT-PACKARD CO. 1981 THU, JUN 4, 1981, 2:59 PM  
 >

# VERIFY

In the final example, the VERIFY PREP option is demonstrated again, this time with two additional parameters, PMAP and MAXDATA. The PMAP parameter produces a descriptive listing of the file on the screen (1). Following the listing, an execution message informs the user that the segmenting is completed (2). The temporary file is made permanent with the use of the MPE :SAVE command (3), and :LISTF,2 again brings descriptive file information to the screen (4). PROGFILE is then RUN, and the calculator is displayed and applied (5).

```
>V P PRDGFIL;PMAP;MAXDATA=10000
(1) HP32050A.01.03 SEGMENTER/3000 (C) HEWLETT-PACKARD CO. 1979
--
PROGRAM FILE PRDGFIL.RPG.SUBSYS

RPG'01          0
NAME           STT  CODE ENTRY SEG
RPG'DB         1    0    0
R'DUT         2
R'INT         3
R'CALT        4
R'CALD        5
R'CNTRL       6
SEGMENT LENGTH      34

RPG'00          1
NAME           STT  CODE ENTRY SEG
R'DUT         1    0    0
R'EDIT        6
R'LINEC       7
R'CALT        2  574  574
R'CALD        3  600  600
DIVD         10
R'INT         4  766  766
R'ERRDR      11
I'0001        5  766 1052
SEGMENT LENGTH     1130

PRIMARY DB      134  INITIAL STACK  1440  CAPABILITY      600
SECONDARY DB   277  INITIAL DL      0    TOTAL CODE     1164
TOTAL DB       433  MAXIMUM DATA 23420  TOTAL RECORDS  14
ELAPSED TIME   00:00:04.673  PROCESSOR TIME 00:00.576
-
```

(2) Segmenter completed.

(3) >:SAVE PRDGFIL

(example continued)

(4) >:LISTF PROGFILE,2  
 ACCOUNT= SUBSYS           GRDUP= RPG

FILENAME	CODE	-----LOGICAL RECDR-----		-----SPACE-----		
		SIZE	TYP	EOF	LIMIT R/B	SECTORS #X MX
PROGFILE	PROG	128W	FB	12	12 1	13 1 1

>RUN PROGFILE

(5)       SIMPLE                   CALCULATR

ENTER DATA IN THE FORMAT:  
 C NNNDD NNNDD  
 S 22222 11111

..... YOUR PROBLEM: 222.22 SUB 111.11 EQUALS       111.1100

ENTER DATA IN THE FORMAT:  
 C NNNDD NNNDD  
 :

HP#####X.00.00 RPG INTERACTIVE SYSTEM ENVIRONMENT RISE/3000  
 (C) HEWLETT-PACKARD CO. 1981           THU, JUN 4, 1981, 3:02 PM

>

# XPAND

XPAND expands file limit size of the current work file.

## Form

X [PAND] numrecs

## Parameter explanation

numrecs        Number of records by which you wish to expand the file size.

## Purpose

The XPAND command increases the file limit size of the current work file by a specified number of records.

## Related commands

When you use the FILE command to create a new file, RISE asks you for the maximum number of records the file will contain. If, for example, you specified 10 lines for the new file, entered 10, and discovered that you had more lines to add to the file, RISE would return the following message:

Error 147: No room in edit file; use XPAND.

Using X [PAND] with the number of records you wish to add, you could enlarge the size of the file as desired. RISE would respond with the following message, when the expansion is accomplished:

Xpand completed.

Execution mode: Line



## EXAMPLES

The following is a legal abbreviation which may be used with the XPAND command and its parameters:

X100 Increase the file limit size of the current work file by 100 records.

In the example, a KSAM file is created with a maximum number of 3 records (1). An attempt to LIST the complete contents of the file results in an error message because, at this point, the file is empty (2). A second error is caused by an attempt to enter more records into the file than were specified as the maximum number previously (3). The XPAND command is used to expand the maximum number of records the file will accept by ten (4). When the ADD command is again employed, the file accepts the additional records (5).

```

>FILE NEWSIM
  Creating new Ksam file, enter maximum number of records.
(1) 3
    File opened.
>L ALL
(2) Error 31: Range is empty.
>A
    1      :LINE 1
    2      :LINE 2
    3      :LINE 3
    4      :LINE 4
(3) Error 147: No room in edit file, use XPAND.
>L A
    1      LINE 1
    2      LINE 2
    3      LINE 3
>
(4) >XPAND10
    Xpand completed.
(5) >ADD LAST
    3      LINE 3
    4      :LINE 4
    5      :LINE 5
    6      :<Control Y>
>

```

# :MPEcommand

:MPEcommand executes MPE command.

Form
:MPEcommand

## Purpose

Executes an "MPEcommand" following a colon — : — that can be executed programmatically with RISE. How these commands are executed is described in the MPE Intrinsic Manual under Command Intrinsic.

## Related commands

When you type H [ELP] : a list of the MPE commands which can be executed programatically by RISE will be displayed on the screen.

Execution mode: Line

## EXAMPLES

A list of some MPE commands that may be executed through RISE will appear on the screen when HELP: is entered. Below is a copy of the command display which appears in response to this command.

```

██████████  ██████████  SEQNUM  COMMAND  ██████████  RETURN  LINE  MAIN
* OFF *    WINDOW  TO SHOW  MODE  MENU

: Command:      Execute MPE command.

Syntax:         :MPEcommand

Description:    Executes "MPEcommand" where "MPEcommand" is one of:
                ALTSEC    LISTF      REPORT    SHOWDEV   STREAM
                ALTVSET   LISTVS    RESET     SHOWIN    TELL
                BUILD     NEWVSET   RESETDUMP SHOWJOB   TELLOP
                COMMENT   PTAPE    PESTORE   SHOWJOB
                DSTAT     PURGE     SAVE      SHOWME
                DISLINE   PURGEVSET SECURE    SHOWQOUT
                FILE      RELEASE   SETDUMP   SHOWTIME
                GETPIN    REMOTE_HELLO SETJOB    SPEED
                HELP      PENAME   SETMSG    STORE

Examples:      :SHOWJOB
                :FILE LP:DEV=LP

COMMAND: [████████████████████████████████████████████████████████████████████████████████]
ENTER?
COMMAND MODE
```

In this example, a file equation is entered which specifies file L as a line printer (1). The results of entering :SHOWTIME follow (2). A LISTF,2 provides you with information about the current work file (3). A means of purging a file ends the example (4). Note that, in all the examples, the user provided the colon (:)  
prompt.

(1) >:FILE L;DEV=LP

(2) >:SHOWTIME  
THU, MAY 28, 1981, 3:20 PM  
>

(3) >:LISTF BACKUP,2  
ACCOUNT= SUBSYS GROUP= RPG

FILENAME	CODE	-----	LOGICAL	RECORD	-----	SPACE	-----
		SIZE	TYP	EDF	LIMIT R/B	SECTORS	#X MX
BACKUP		88B	FA	35	38 2	20	7 7

(4) >:PURGE BACKUP  
>  
>



# RECOVERY PROCEDURES

SECTION

III

While using the RPG Interactive System Environment, you may encounter one of six problem situations. They are:

System failures	Terminal lockouts
Breaking out of Block Mode	Transmission errors
Loss of echo at the screen	Effects of :EOD

For your information, RISE uses the following file designators during operation:

*Wdddhhmm* — KSAM data file for work file  
*WKdddhhmm* — KSAM key file for work file  
*Ldddhhmm* — compilation listing file  
*Udddhhmm* — Undelete file (in temporary domain)

If RISE abnormally terminates, because of system failure, use of the Break key, or entering :EOD, you may find these files are permanent in your group and account. The KSAM work file should be recovered, and the compilation listing file and the UNDELeTe file can be purged.

## A. Recovering from System Failures

If the system fails while you are editing a copy of a file (initiated by the TEXT command), do the following:

1. Determine the designator of the file or files placed in jeopardy by the failure.
  - a. the name of the work file, which is a permanent KSAM file, is in the form *Wdddhhmm*, where “ddd” is the day of the year, “hh” is the hour of the day, and “mm” is the minute of the hour.
  - b. the name of the associated key file is *WKdddhhmm*, where K overlays the first “d” in day of the year, and “dddhhmm” have the same meaning as they do with the work file.

- c. perform a :LISTF W@#####2 to determine the designators of the files.
2. After you have the name of the work file, enter the following command:

```
:RUN KSAMUTIL.PUB.SYS
```

3. When you receive a ">" prompt at the terminal enter the following two commands to KSAMUTIL:

```
>KEYINFO filename;RECOVER
```

(and, after recovery is made)

```
>EXIT
```

(In the above, "filename" is the name of the KSAM data file determined in the LISTF command.)

This will recover the work file and take you out of KSAMUTIL. You can then run RISE and continue editing after you TEXT in the recovered work file name.

If the system fails while you are editing the file directly (initiated by the FILE command), perform steps 2 and 3 above. In other words, use KSAMUTIL and the KEYINFO command with the RECOVER option with the name of the direct file to restore it for continued editing.

## B. Recovery from Terminal Lockout

Terminal lockout is the term used to describe a state in which the terminal is totally unresponsive to any key you press. To recover, do the following:

1. If you are in Line Mode, just press the RESET TERMINAL key twice. This is a hard reset.
2. If you are in Block Mode, press the RESET TERMINAL key twice, for a hard reset. Then press softkey F7 to enter Line Mode. An exception occurs when you are in Scroll set. If you are in Scroll set (SHOW mode), the special function key F7 has a different function. In order to return to Line Mode, you must first press F1, which will take you out of Scroll set and into Master set. You may then press F7 to enter Line Mode.

## C. Recovery from Breaking out of Block Mode

If you hit the BREAK key inadvertently while in Block Mode, do the following:

1. Do a hard reset (press the RESET TERMINAL key twice).
2. Type "ESC:" to turn echo on. (At this point you should be able to communicate with MPE.)
3. Type RESUME to resume execution of the editor. MPE will respond with "Read Pending."
4. Enter the carriage return.
5. If you are in Scroll set (SHOW mode), press F1 to reset the special function keys to Master set.
6. Press softkey F8 to display the main menu.
7. Press softkey F8 to refresh the main menu. This will return you to Block Mode.

Note: Steps 6 and 7 may be skipped if you press softkey F7, which puts you in Line Mode instead of Block Mode.

## D. Recovery from a Transmission Error

Transmission errors usually bring to the screen a nonsensical clutter of characters. The keyboard works but characters, such as escape characters, appear on the screen erroneously. They cause incorrect displays of forms in block mode.

To recover from the effects of transmission errors do the following:

1. If you are in Scroll set (SHOW mode), press F1 to reset the special function keys to normal SHOW mode functions (Master set).
2. Press F8 to display the main menu.
3. Press F8 a second time to refresh the main menu.

You can also press F7 to enter Line Mode and re-enter your command.

## E. Recovering echo after it is lost

When your screen remains blank everytime you type a key, your terminal may have lost its echo. You can recover it in one easy step.

1. Press the ESCAPE key followed by the colon (:).

## F. Recovery from :EOD

If you accidentally or intentionally enter :EOD (End of Data) while in Line mode, RISE will immediately terminate. Your KSAM work file will not be purged.

If you were editing a copy of a permanent file (TEXT) or had created a new work file (ADD), you can locate the work file by typing the following:

```
:LISTF W@#####,2
```

See Section A, above, for a discussion of this file designator.

To recover this work file, you simply TEXT in the KSAM data file and KEEP it to a sequential permanent file. Then you use the KSAMUTIL to purge the KSAM work file with the name located using :LISTF as described above.

If you entered :EOD while editing a file directly (FILE), no harm is done and the :EOD operates like the EXIT command. Your changes to the file will be permanent.



# COMMAND SUMMARY

APPENDIX

A

## 1. A [DD] [lb] [BY inc] [^^ [form] [R [EPEAT] [n] ] ]

ADD lines starting at "lb" BY inc (rement); "^^" controls the RPG Record Spec forms and may be entered while in Add Mode at the start of a new line; the "R [EPEAT] n" option homes the cursor and clears the screen after "n" number of lines have been added. The LIST, DELETE, and MODIFY commands may be entered while in the Add Mode by typing ">>" at the start of a new line followed by the command.

Also, you can type ">>" alone and RISE will prompt you for text starting at column 6 instead of column 1. The command automatically skips over the RPG line numbers in column 1-5. You type "<<" to get back to column 1.

## 2. B [EGIN] [N [OW] ]

End current editing session to start editing a new file as if you had just entered RISE; if editing session was initiated by FILE command, close the direct work file to keep the changes; if editing session was initiated by TEXT command, display safety messages before purging the work file; the N [OW] option disables the safety messages and clears the file, initiated by the TEXT command, immediately.

## 3. CH [ANGE] "oldstring" TO "newstring" [IN lb [/le] ]

Changes "oldstring" to "newstring", starting at "lb" and ending at "le".

## 4. COM [MENT] [C] [lb] [BY inc]

Displays a comment banner 10 lines long with the RPG Record Spec form of the type corresponding to the line before "lb", where the banner will appear; "C" increases the banner size to 15 lines and displays a column indicator; "BY inc" establishes the increment value for each line within the banner.

## 5. COP [Y] lb [/le] TO loc [BY inc]

Duplicates lines starting at "lb" TO a specified "loc"ation; "lb/le" establishes a range of lines to be copied; "BY inc" sets the increment value for each line being copied.

## 6. D [ELETE] lb [/le]

Delete line or lines starting at "lb"; stop the deletion after "le".

7. E [XIT] [N [OW]]

Ends the subsystem; "NOW" parameter establishes that the command is to be executed immediately, without RISE prompting you with safety messages and without saving the work file with the KEEP command.

8. FIL [E] filename [N [OW]]

Allows you to edit a file directly, without the delays resulting from the TEXT/KEEP commands; the "NOW" parameter used with this command disables safety prompts and will open "filename" even if you have been editing a work file that was created with the TEXT command and you did not keep your changes in a permanent file. The "NOW" parameter immediately discards the current work file.

9. F [IND] [E [VERY]] "string" [IN lb [/le]]

Locates "string"; "EVERY" specifies that every occurrence of "string" is to be located; if "EVERY" is omitted, only the first occurrence is located; you type "F" alone to find the next occurrence; "IN lb/le" restricts the search to a range.

10. FO [RM] form

Displays the RPG Record Spec form of type "form". "form" may be H,F,E,L,I,C,O, or COL.

11. G [ET] filename

Reads and executes commands stored in the file "filename".

12. H [ELP] [command]

Gives brief descriptions of all RISE commands; when "command" is specified, provides detailed descriptions of single command.

13. I [NCR] [value]

Sets default increment value; "value" sets new default increment value which prevails until "I [NCR]" is used again.

14. J [OIN] filename [TO loc] [BY inc]

Appends or merges an entire file named "filename"; "TO loc" locates position where "filename" is to be joined; "BY inc" establishes increment value.

15. K [EEP] [filename] [N [OW]] [U [NN]]

Copies the work file into a file named "filename"; "NOW" parameter instructs RISE to overwrite an existing copy of the file without RISE prompting you with safety messages; "UNN" specifies that the file is kept without appended sequence numbers.

16. LIN [E]

Enters line mode.

17. L [IST] [lb [/le]] [O [FF]]

Lists lines in file; "le/lb" is the range of lines for listing; "OFF" lists the lines offline to a line printer instead of to your terminal with the formal file name of "RISELIST" and device class of "LP".

18. ME [NU]

Displays main menu.

19. M [ODIFY] [lb [/le]] [^^]

Lists line [s] for modification; "lb/le" establishes range of lines for modification; "^^" causes appropriate RPG Record Spec form to appear with the line. (Type <Control Y> to restore record and "/" to exit a range of lines.)

20. MOV [E] lb [/le] TO loc [BY inc]

Moves line (or lines) "lb/le" to a specified location—"TO loc"; "BY inc" parameter establishes increment values.

21. P [RINT] lb [/le] [R [PGLIST]]

Prints line (or lines) offline in formatted form, skipping 3 lines whenever the form type in column 6 changes, with the formal filename of "RISELIST" and device class of "LP"; if "RPGLIST" is specified, the compilation listing is printed offline, instead of the work file, with the formal filename of "RPGLIST".

22. R [ENUM] [lb/le] [BY inc]  
 BOTH [BY inc]  
 SO [URCE] [BY inc]  
 SO [URCE] W [ITH] SE [QNUM]  
 SE [QNUM] W [ITH] SO [URCE]

Renumbers work file sequence line numbers; "lb/le" establishes range for renumbering; "BY inc" establishes increment value; "BOTH" rennumbers sequence line numbers and RPG source line numbers (cols 1-5); "SOURCE WITH SEQNUM" rennumbers source lines to agree with the sequence line numbers; "SEQNUM WITH SOURCE" rennumbers sequence line numbers to agree with source line numbers.

23. RU [N] progfile [,entrypoint] [;NOPRIV] [;LMAP] [;DEBUG]  
 [;MAXDATA=segsz] [;PARM=num]  
 [;STACK=stacksize] [;DL=dlsz]  
 P  
 [;LIB=G] [;NOCB]  
 S

Runs a program file (similar to MPE :RUN command); see RUN command for an explanation of parameters.

24. S [HOW] [lb] [C [OL]]

Shows a page for direct screen modification with RPG Record Spec form appropriate to records displayed; "COL" parameter displays a column indicator rather than the RPG Record Spec form.

25. T [EXT] filename [N [OW]]

Copies the file "filename" into a work file; "NOW" allows you to TEXT in a new work file without keeping the changes you have made to the work file you are editing and without RISE prompting you with a safety message.

26. UN [DEL]

Restores lines deleted by the last "DELETE" command.

27. V [ERIFY] R [PG]  
 L [IST]  
 P [REP] [progfile] [;ZERODB] [;PMAP]  
 [;MAXDATA=segsz] [;STACK=stacksize]  
 [;DL=dlsz] [;CAP=caplist] [;RL=filename]  
 [;PATCH=patchsize]

"RPG" compile the edit file; or "LIST" a previously produced compilation listing. "PREP" prepares a USL file in \$OLDPASS to a program with the preparation "options" that are described under VERIFY.

28. X [PAND] numrecs

Expands the size of the work file by "numrecs" (number of records).

29. :MPEcommand

Executes an "MPEcommand" entered following a colon (:) character. The MPE commands which you may enter are listed when you type H [ELP] .:



# COMPARISON OF RISE AND EDIT/3000

APPENDIX

B

RISE and Edit/3000 are in many respects similar editors. There are significant differences between them, however. Below is a list of contrasting features that new RISE users, who are accustomed to Edit/3000, will find helpful.

RISE/3000 Features	Edit/3000 Features
1. Line and Block Modes	Line mode only
2. No delimiters between parameters	Delimiters between parameters
3. Only capital letters accepted	Capital and lower case letters accepted
4. KSAM work file	Extra data segments in addition to a work file.
5. Only 1 command accepted at a time	Multiple commands allowed
6. No quiet mode	Optional quiet mode
7. UNDEL command to recover deletions	No delete recovery
8. PRINT ALL prints offline with spacing between changes in column 6 (form type). LIST ALL OFF prints offline without spacing.	LIST ALL, OFFLINE prints offline
9. HELP command replaces	XPLAIN command
10. MOVE command replaces	GATHER command
11. RENUM command replaces	GATHER ALL command
12. GET command replaces	USE command
13. Command must be completed on 1 line (can't exceed 50 characters in length)	Ampersand ( & ) used to continue a command on next line







# INFORMATIVE EXECUTION MESSAGES

APPENDIX

C

---

## NUM Informative Execution Message

---

0	Syntax checking is turned on.
1	Syntax checking is turned off.
2	Preparing for compilation...
3	Return to main menu with the MENU command, then compile.
4	Text completed.
5	Warning: record size is greater than 88 bytes, truncated to 80.
6	File is unnumbered; appending sequence numbers.
7	Print completed.
8	Increment reset to 1.
9	Increment set.
10	No errors found.
11	At End-Of-File.
12	No compilation errors occurred.
13	Error message not found.
14	Position cursor at line to add after, then press softkey F1-F4.
15	Insert lines cancelled.
16	No lines to show; displaying main menu.
17	Keep completed.
18	Delete completed.
19	Can't Un-delete a line; the line number exists already.
20	Undel completed.
21	'String' not found.
22	String found.
23	Changes completed.
24	No changes were made.
25	Entering Line mode - use Carriage Return.
26	Entering Block Mode - use Enter Key.
27	TITLE
28	Xpand completed.
29	Creating new Ksam file; enter maximum number of records.
30	Enter response and press ENTER key.
31	Errors in response; Command cancelled.
32	File opened.
33	Enter new file name followed by additional number of records.

---

## NUM Informative Execution Message

---

34 File exists already; enter a NEW file name.  
35 Join completed.  
36 Copy completed.  
37 Move completed.  
38 Renum completed.  
39 Get completed.  
40 Creating work space file.  
41 Keep cancelled.  
42 End-Of-Data.  
43 KEEP not done. CLEAR current work file?  
44 File cancelled.  
45 Text cancelled.  
46 Exit cancelled.  
47 Offline listing completed.  
48 Segmenter completed.  
49 Ready to begin new session.  
50 Begin cancelled.

# ERROR MESSAGES

---

## NUM Error Message

---

0	Can't sense cursor position.
1	Only one command allowed.
2	Incorrect specification of the command in the menu.
3	Only F,T,K,J allowed in File type command.
4	Only I,P,C,R,M allowed in Gen1 type command.
5	Only D,S,A,F,CO,CH allowed in Edit type command.
6	Only U,H,V,E allowed in Gen2 type command.
7	There is no such editor command.
8	Syntax error on ADD command.
9	Syntax error on CHANGE command.
10	Syntax error on COMMENT command.
11	Syntax error on COPY command.
12	Can't distinguish between COMment or COPy command.
13	Can't distinguish CHange, COMment, or COPy command.
14	Syntax error on DELETE command.
15	Syntax error on EXIT command.
16	Can't recognize the command.
17	Syntax error on FILE command.
18	Syntax error on FIND command.
19	Can't distinguish between FILE, FINd, or FOrm.
20	Syntax error on HELP command.
21	Syntax error on INCR command.
22	Syntax error on JOIN command.
23	Syntax error on KEEP command.
24	Syntax error on MENU command.
25	Syntax error on MOVE command.
26	Can't distinguish MEnu, MOVe, or MODify command.
27	Syntax error on PRINT command.
28	Syntax error on RENUM command.
29	Syntax error on SHOW command.
30	Syntax error on TEXT command.
31	Syntax error on UNDEL command.
32	Syntax error on VERIFY command.
33	Syntax error on NOW parameter.
34	Only NOW parameter allowed with Exit command.
35	KEEP not done - use NOW option to CLEAR work file.
36	Illegal special character.
37	
38	Syntax error on RPG option.
39	Syntax error on ON option.
40	Syntax error on OFF option.
41	Can't distinguish between ON or Off.
42	Only one of RPG, LIST, or PREP allowed.

---

## NUM Error Message

---

- 43 Need Process Handling Capability.
- 44 There is no file to compile.
- 45 Unable to create son process for compilation.
- 46 Unable to activate RPG compiler.
- 47 Unable to set up File equations for compilation.
- 48 Unable to Build RPGLIST file for compilation.
- 49 Syntax error on LIST option.
- 50 There is no compilation listing; use the RPG option.
- 51 Increment too big; line numbers exceed limit 99999.999.
- 52 There are no records in the file.
- 53 A numbered file has illegal characters as line numbers.
- 54 A file name is required with this command.
- 55 A file name may only contain a max. of 37 characters.
- 56 Filename must start with letter followed by letter/num.
- 57 Only NOW option allowed with this command.
- 58 Can't edit this type of file, only RPG source programs.
- 59 A line number is required.
- 60 A line number is required before the RPGLIST option.
- 61 Only a '\*',line number,FIRST,LAST,ALL allowed in range.
- 62 The TO line cannot be less than the FROM line number.
- 63 Only the RPGLIST option is allowed after the range.
- 64 There is no compilation listing to print.
- 65 There is no edit file.
- 66 Syntax error on Rpglist option.
- 67 Syntax error on All option.
- 68 A special delimiter may not follow the line range.
- 69 Illegal delimiter.
- 70 Syntax error on First.
- 71 Syntax error on Last.
- 72 A maximum of 5 digits is allowed left of decimal.
- 73 Range is empty.
- 74 A maximum of 3 digits is allowed right of decimal.
- 75 A number is required to right or left of decimal.
- 76 Illegal numeric increment value.
- 77 Illegal parameters in command.
- 78 An increment value cannot be zero.
- 79 Syntax error on COL option.
- 80 No parameters allowed after COL option.
- 81 Range is empty.
- 82 No room in file; use XPAND to expand file.
- 83 Can't insert a new line.
- 84 New insert line number exceeds 99999.999.
- 85 No room between lines to insert new line number.
- 86 Cursor must be pointing at a line.
- 87 Blanks may only follow the UNN option.
- 88 Syntax error on UNN option.
- 89 No edit file name for default; give Keep file name.

---

## NUM Error Message

---

90	File exists already, use NOW option to replace it.
91	Only NOW and/or UNN option allowed.
92	No lines were Deleted to Un-delete.
93	No parameters are allowed with this command.
94	No string to find.
95	Missing range after IN option.
96	Only a delimited string and/or IN range allowed.
97	Missing close delimiter.
98	A string must be given between the delimiters.
99	A slash (/) cannot delimit a string.
100	Should be blank after line range.
101	'oldstring' TO 'newstring' missing.
102	Oldstring & newstring must be enclosed by delimiters.
103	TO missing after 'oldstring'.
104	Only IN range allowed after 'newstring'.
105	Missing close delimiter for 'oldstring'.
106	An oldstring must be specified between delimiters.
107	Missing close delimiter for 'newstring'.
108	A newstring must be specified between delimiters.
109	Oldstring length must equal newstring length.
110	Syntax error on LINE command.
111	Syntax error on LIST command.
112	Can't distinguish between LINE or LIST command.
113	Missing MPE command.
114	Undefined MPE command.
115	Could not execute this command.
116	Can't keep to a KSAM file — only sequential file.
117	Syntax error on RUN command.
118	Can't distinguish between REnum and RUn commands.
119	Missing program name.
120	Unable to create process.
121	Missing secondary entry point after comma.
122	Expecting parameter after semi-colon.
123	Illegal RUN parameter.
124	Can't activate process.
125	Expecting equal sign after a parameter.
126	Illegal numeric value after the equal sign.
127	Numeric value may only have a maximum of 5 digits.
128	Only G,P,S allowed for libsearch value.
129	Program name may contain a maximum of 8 chars.
130	Expecting lockword immediately after the slash.
131	Lockword may contain a maximum of 8 characters.
132	Expecting group name immediately after period.
133	Group name may contain a maximum of 8 chars.
134	Expecting account name immediately after period.
135	Account name may contain a maximum of 8 chars.
136	Syntax error on XPAND command.

---

## NUM Error Message

---

137	Missing number of records to expand by.
138	Illegal numeric value.
139	Expansion value must be greater than zero.
140	Can't determine a unique temporary file name.
141	Only BY increment allowed after TO.
142	Only TO line and/or BY increment allowed.
143	Internal error — could not convert to double value.
144	No room between line numbers.
145	Missing or illegal line number.
146	Missing numeric increment after BY.
147	No room in edit file; use XPAND.
148	Missing line range.
149	TO location missing.
150	Only BY increment allowed after TO location.
151	TO location cannot be within line range.
152	Only BY increment allowed after range.
153	Increment cannot be less than 1 for source.
154	Only BY increment allowed after BOTH.
155	Syntax error on BY or BOTH.
156	BY increment cannot be greater than 5000.
157	Expecting SOURCE after WITH.
158	Expecting WITH SOURCE after SEQNUM.
159	Expecting SEQNUM after WITH.
160	Only WITH or BY allowed after SOURCE.
161	Syntax error on SOURCE.
162	Syntax error on SEQNUM.
163	Can't distinguish between SOURCE and SEqnum.
164	Syntax error on WITH.
165	A source line number has non-numeric characters.
166	A source line number is not in ascending sequence.
167	No room between lines to insert a page.
168	Syntax error on Get command.
169	Can't execute Get command from a file.
170	Only 0-8 allowed for a softkey from a file.
171	Pause length must be 0 to 64000.
172	Syntax error on FOrM.
173	Only H,F,E,L,I,C,O, or COI allowed.
174	Missing or illegal offset integer after *.
175	Expecting ^^.
176	Only H,F,E,L,I,C,O,COL and/or Repeat allowed after ^^.
177	Syntax error on BY.
178	Syntax error on Repeat.
179	Illegal integer value after Repeat.
180	Unexpected error — read from terminal failed!
181	Line truncated to 80 characters.
182	Only DELETE, LIST, or MODIFY allowed in Add mode.
183	FATAL ERROR in Show Mode, returning to Line Mode.

---

## NUM Error Message

---

184	Line would exceed 80 characters.
185	Syntax error on MODIFY command.
186	Can't distinguish MODify and MOVE commands.
187	A subcommand may only be entered in columns 1-80.
188	Should be blank after Repeat option.
189	Should be blank after BY increment.
190	Should be blank after file name.
191	Should be blank after ^^.
192	Only ^^ allowed after line range.
193	Should be blank after SOURCE.
194	Should be blank after SEQNUM.
195	Only capital letters are allowed.
196	Syntax error on EVERY.
197	Should be blank after OFF.
198	Only OFF allowed after line range.
199	Syntax error on PREP.
200	No USL in \$OLDPASS, use RPG option to compile.
201	Could not activate Segmenter.
202	\$OLDPASS is only allowable system file name.





- a**
- Accessing files, 1-13
  - ADD, 2-8
    - leaving ADD mode, 2-9
    - with LIST, MODIFY, and DELETE commands, 2-9
- b**
- BEGIN, 2-14
  - Block mode, advantages, 1-5
    - features, 1-8
  - Bracketing values for specification forms, see RENUM, 2-81
- c**
- CHANGE, 2-19
    - delimiters used with, 2-19
  - Command menu, discussion, 1-8
    - use in show mode, 1-10
  - Command window, discussion, 1-9
  - COMMANDS, general information, 1-17
    - divided by function on main menu, 1-20
    - abbreviating, 1-17
    - limitations, 1-21
    - listed with legal abbreviations, 1-18
    - notations, 2-1
    - separating parameters, 1-21
    - summary, A-1
    - used in block mode only, 1-19
    - used in either mode, 1-19
    - used in line mode only, 1-19
  - COMMANDS, detailed discussions, 2-1
    - ADD, 2-8
    - BEGIN, 2-14
    - CHANGE, 2-19
    - COMMENT, 2-21
    - COPY, 2-24
    - DELETE, 2-26
    - EXIT, 2-28
    - FILE, 2-30
    - FIND, 2-39
    - FORM, 2-42
    - GET, 2-45
    - HELP, 2-49
    - INCRement, 2-52
    - JOIN, 2-55
    - KEEP, 2-58
    - LINE, 2-62
    - LIST, 2-63
    - MENU, 2-67
    - MODIFY, 2-70
    - MOVE, 2-76
    - MPEcommand, 2-124
    - PRINT, 2-79
    - RENUM, 2-81
    - RUN, 2-86
    - SHOW, 2-91
    - TEXT, 2-104
    - UNDELeTe, 2-108
    - VERIFY, 2-111
    - XPAND, 2-122
  - COMMENT, 2-21
    - deleting comment in show mode, 2-21
  - COPY, 2-24
    - compared to MOVE, 2-76
  - Comparison of RISE/3000 and EDIT/3000, B-1
  - Control Y, used to cancel modifications, 2-72
    - effective in LINE mode only, 2-2
    - halts execution of LIST, 2-63
    - used to leave ADD mode, 2-9
- d**
- DELETE, 2-26
  - Deleting comments, 2-21
  - Delimiters, with CHANGE, 2-19
    - with FIND, 2-39
    - with RUN, 2-86
- e**
- EDIT/3000 and RISE/3000 compared, B-1
  - Error messages, D-1
  - Execution messages, C-1
  - EXIT, 2-28

# INDEX (continued)

## f

FILE, 2-30  
  used in creating new file, 2-31  
  used in accessing old KSAM file, 2-31  
  used in converting non-KSAM to new KSAM file, 2-32  
  leaving FILE, 2-33  
File designators for RISE/3000, 3-1  
File names, reserved letters, 2-58  
FILE/EXIT versus TEXT/KEEP, see accessing files, 1-13  
Files, accessing, 1-13  
  KSAM files, 1-12, 2-33, 3-1  
  permanent, 1-13  
  safeguards, 1-14  
  work, 1-12  
  discussion, 1-12  
FIND, 2-39  
  delimiters used with command, 2-39  
FORM, 2-42

## g

GET, 2-45  
  function keys used with command, 2-45

## h

Hardware requirements, 1-3  
HELP, 2-49  
  function keys used with command, 2-49  
HELP:, use with MPEcommand, 2-124  
How to run RISE/3000, 1-15

## i

INCRement, 2-52  
  limitations, 2-52  
Increment values, general discussion, 2-4

## j

JOIN, 2-55

## k

KEEP, 2-58  
KSAM, purging files, 2-33  
  work files, 1-12  
  use in recovery procedures, 3-1  
KSAM/3000, 1-3

## l

Limitations on commands, 1-21  
Limits of increment values, 2-4  
LINE, 2-62  
Line mode, advantages, 1-5  
Lines, renumbering, see RENUM, 2-81  
LIST, 2-63  
  compared to PRINT, 2-63  
Literal parameters, 2-1

## m

Master set in show mode, 2-92  
Maximum characters in a command, 1-21  
MENU, 2-67  
  function keys with main menu, 2-67  
Modes, summary of line and block modes, 1-7  
Modes of operation, 1-5  
MODIFY, 2-70  
  effects of control-y, 2-72  
  leaving MODIFY mode, 2-72  
  subcommands, 2-70  
MOVE, 2-76  
  compared to COPY, 2-76  
MPE IV, 1-3  
MPEcommand, 2-124  
  use of HELP:, 2-124

## n

Notations for commands, 2-1  
Now parameter, general discussion, 1-14

## p

Parameters, correct format in commands, 1-21  
    how to separate parameters, 1-21  
    literal, 2-1  
Permanent files, 1-13  
PRINT, 2-79  
    compared to LIST, 2-79

## r

Record pointer, description, 2-3  
Record specification forms, 1-4  
    see FORM, 2-42  
Recovery procedures, 3-1  
    echo after it is lost, 3-4  
    from system failures, 3-1  
    from :EOD, 3-4  
    from a transmission error, 3-3  
    from breaking out of block mode, 3-3  
    from terminal lockout, 3-2  
RENUM, 2-81  
Requirements, hardware, 1-3  
Requirements, software, 1-3  
RISE/3000 file designators, 3-1  
RISEtour, 1-15  
RPG specification forms, 1-4  
    bracketing values for forms, see RENUM, 2-81  
RUN, 2-86  
    semi-colon (;) required as delimiter, 2-86  
Running RISE/3000, 1-15

## s

Safeguarding files, discussion, 1-14  
Scroll set in show mode, 2-93  
Sequence numbers, relative specifications, 2-3  
    specifying increments, 2-4  
Show mode, discussion, 1-9  
SHOW, 2-91  
    master set discussion, 2-92  
    scroll set discussion, 2-93  
SIMCAL (simple calculator), 2-5  
    a listing of the file, 2-5  
    compiling, preparing, running, 2-6  
    used with RISEtour, 1-16  
Sequence numbers, 2-4, 2-52

Softkeys, see special function keys, 1-11  
Software requirements, 1-3  
Special function keys, discussion, 1-11  
    with HELP, 2-49  
    with MENU, 2-67  
    with SHOW, master set, 2-92  
    with SHOW, scroll set, 2-94  
    with VERIFY, 2-114  
Specifying sequence numbers, 2-3  
Split screen, use with VERIFY command, 2-114  
Subcommands with MODIFY, 2-70  
Summary of commands, A-1

## t

TEXT, 2-104  
TEXT/KEEP and FILE/EXIT compared, 1-13

## u

UNDELETE, 2-108  
Undeleting in SHOW mode, 2-108

## v

VERIFY, 2-111  
    special function keys with command, 2-114  
VPLUS/3000, 1-3

## w

Work file, 1-12  
    naming, 1-12

## x

XPAND, 2-122



**READER COMMENT SHEET**

**HP 3000 Computer Systems  
Utilities Reference Manual  
(RPG Interactive System Environment - RISE/3000)**

32033-90068

July 1986

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?      Yes  No  (If no, explain under Comments, below.)

Are the concepts and wording easy to understand?      Yes  No  (If no, explain under Comments, below.)

Is the format of this manual convenient in size, arrangement, and readability?      Yes  No  (If no, explain or suggest improvements under Comments, below.)

Comments:



**FROM:**

**Name** \_\_\_\_\_

**Company** \_\_\_\_\_

**Address** \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

FOLD

FOLD



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 1070 CUPERTINO,CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager  
Hewlett-Packard Company  
General Systems Division  
19447 Pruneridge Avenue  
Cupertino, California 95014



FOLD

FOLD