

**UNISYS**

A Series  
SORT Language  
**Programming  
Reference Manual**

Release 3.8.0

Priced Item

May 1989  
Distribution Code SE  
Printed in U S America  
1169794.380

**UNISYS**

**A Series  
SORT Language  
Programming  
Reference Manual**

Copyright © 1989 Unisys Corporation  
All rights reserved.  
Unisys is a trademark of Unisys Corporation

Release 3.8.0  
Priced Item

May 1989  
Distribution Code SE  
Printed in U.S.A.  
1169794.380

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

**NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT.** Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded using the Remarks form at the back of this manual, or may be addressed directly to Unisys, Technical Publications, 460 Sierra Madre Villa, Pasadena, CA 91109



# Product Information Announcement

New Release    Revision    Update    New Mail Code

---

Title

**A Series SORT Language Programming Reference Manual**

This Product Information Announcement announces the release of Update 2 to the *A Series SORT Language Programming Reference Manual*, dated May 1989, relative to the Mark 3.8.0 System Software Release.

This manual describes the details of designing, coding, and running Sort programs using the SORT language.

Update 2 adds the instructions for the COLLATE statement. The COLLATE statement is a new optional SORT statement. Update 2 also corrects the examples for sorting by integer.

Changes to the text are indicated by vertical bars in the margins of the replacement pages.

**Remove**

iii through xvi  
3-1 through 3-2  
4-1 through 4-2  
  
4-3 through 4-4  
A-1 through A-2  
C-1 through C-2  
D-5 through D-6  
E-3 through E-4  
E-9 through E-12  
E-49 through E-50  
Bibliography-1 through 2  
Index-1 through 22

**Insert**

iii through xvi  
3-1 through 3-2  
4-1 through 4-2  
4-2A through 4-2B  
4-3 through 4-4  
A-1 through A-2  
C-1 through C-2  
D-5 through D-6  
E-3 through E-4  
E-9 through E-12  
E-49 through E-50  
Bibliography-1 through 2  
Index-1 through 22

Retain this Product Information Announcement as a record of changes made to the basic publication.

To order additional copies of these manuals,

- United States customers call Unisys Direct at 1-800-448-1424.
- All other customers contact your Unisys Subsidiary Librarian.
- Unisys personnel use the Electronic Literature Ordering (ELO) system.

To receive the update package only, order 1169794-002. To receive the complete guide, order 1169794.380.

---

Announcement only:

Announcement and attachments:  
AS207

System:     A Series  
Release:    3.9.0 September 1991

Part Number: 1169794-002



Printed on recycled paper



# Page Status

Page	Issue
iii through ix	-002
x	Blank
xi through xv	-002
xvi	Blank
xvii	.380
xviii	Blank
xix	.380
xx	Blank
1-1 through 1-6	.380
2-1 through 2-13	.380
2-14	Blank
3-1 through 3-2	-002
3-3 through 3-5	.380
3-6	Blank
4-1 through 4-2A	-002
4-2B	Blank
4-3 through 4-4	-002
4-5 through 4-13	.380
4-14	Blank
5-1 through 5-13	.380
5-14	Blank
A-1 through A-2	-002
A-3 through A-4	.380
B-1 through B-2	.380
C-1 through C-2	-002
D-1 through D-4	.380
D-5 through D-6	-002
D-7 through D-11	.380
D-12	Blank
E-1 through E-2	.380
E-3 through E-4	-002
E-5 through E-8	.380
E-9 through E-12	-002
E-13 through E-48	.380
E-49 through E-50	-002
E-51	.380
E-52	Blank
F-1 through F-11	.380
F-12	Blank
G-1 through G-9	.380
G-10	Blank

## Page Status

---

Glossary-1 through 9	.380
Glossary-10	Blank
Bibliography-1	-002
Bibliography-2	Blank
Index-1 through 22	-002

# Page Status

Page	Issue
iii through xx	Original
1-1 through 1-6	Original
2-1 through 2-13	Original
2-14	Blank
3-1 through 3-5	Original
3-6	Blank
4-1 through 4-13	Original
4-14	Blank
5-1 through 5-13	Original
5-14	Blank
A-1 through A-4	Original
B-1 through B-2	Original
C-1 through C-2	Original
D-1 through D-11	Original
D-12	Blank
E-1 through E-51	Original
E-52	Blank
F-1 through F-11	Original
F-12	Blank
G-1 through G-9	Original
G-10	Blank
Glossary-1 through 9	Original
Glossary-10	Blank
Bibliography-1	Original
Bibliography-2	Blank
Index-1 through 22	Original





# About This Manual

## Purpose

This manual describes the details of designing, coding, and running Sort programs using the SORT language.

## Scope

This manual concentrates on the components, syntax, and semantics of the SORT language, as well as how to create, compile, and run Sort programs.

## Audience

This manual is intended primarily for applications programmers and individuals with backgrounds in advanced-level programming who want to sort files. However, the text and examples should make it possible for a novice programmer to write, compile, and run a Sort program.

## Prerequisites

A working knowledge of the MCP *SORT* procedure is helpful in understanding the information presented in this manual.

A user should know how to use the A Series Editor and the Command and Edit (CANDE) Message Control System (MCS) to enter code. The user should also know how to use CANDE and the Work Flow Language (WFL) to compile and run jobs.

## How to Use This Manual

This manual has been designed both as a learning text for users who are unfamiliar with the SORT language or B 1000 Series Sort programs and as a reference document for experienced users.

This manual is structured so that users can select and read only those sections meeting their specific needs.

## About This Manual

---

- Users who are unfamiliar with the A Series SORT language and who will be working on A Series systems should read the entire manual except for the “B 1000 SORT Conversion” appendix.
- Users who are familiar with B 1000 Series Sort programs but unfamiliar with the A Series SORT language will find it useful to refer to the “B 1000 SORT Conversion” appendix before reading other sections of this manual.
- Users who are familiar with the A Series SORT language will find this manual most helpful as a reference manual.
- Users who are unfamiliar with railroad diagrams (a Unisys method of depicting syntax) should refer to “Understanding Railroad Diagrams” before reading the body of the manual.

The following conventions are used in this manual:

- Throughout this manual, *A Series* is used to refer collectively to A Series and B 7900 systems.
- In references to the SORT language itself, the SORT procedure, and the SORT compiler, the word *SORT* is in uppercase.
- In references to a program produced using the SORT language, such as a Sort program or Sort job, the word *Sort* has the initial letter capitalized.
- Programs and jobs coded to run with the GSORT compiler control record set to TRUE are identified as GSORT programs and GSORT jobs.

Unless otherwise noted, all references to related documentation within the text is for A Series product information.

## Organization

This manual contains five sections and six appendixes. In addition, a glossary, bibliography, and index appear at the end of the manual.

### Section 1. Creating and Compiling Sort Programs

This section contains both explanations and examples of how to create Sort programs, invoke the SORT compiler from both WFL and CANDE, and run compiled Sort programs from WFL and CANDE.

### Section 2. Sort Program Structure

The overall structure of a Sort program is described and diagrammed in this section. In addition, the basic language components and Sort language keywords are presented.

### **Section 3. SORT Statements**

This section describes, syntactically and semantically, the required SORT statements and their associated attributes.

### **Section 4. Optional SORT Statements**

SORT statements that are optional to the Sort program but useful for improving efficiency are described syntactically and semantically in this section.

### **Section 5. SORT Compiler Control Records**

The use of compiler control records (also referred to as CCRs) within a Sort program are discussed in this section. The compiler options recognized by the SORT compiler are described syntactically and semantically. The GSORT compiler control record is introduced here, but detailed in its own appendix.

### **Appendix A. SORT Language Keywords**

Appendix A lists all the keywords recognized by the SORT compiler, including accepted keyword abbreviations and indication of compatibility with B 1000 Series systems.

### **Appendix B. SORT Compiler Files**

Appendix B describes the various files used by the SORT compiler.

### **Appendix C. SORT Program Files**

Appendix C describes the files that a compiled Sort program can use.

### **Appendix D. B 1000 SORT Conversion**

Sort programs written for Unisys B 1000 Series systems can be easily converted to the Unisys A Series systems SORT language. Appendix D details the changes and compatibility between the two series of systems.

### **Appendix E. Using GSORT to Code and Run Sort Programs**

Appendix E covers the differences in designing, coding, and executing Sort programs that use the GSORT option fixed-format specifications and statement.

### **Appendix F. Using the Standard Collating Sequence**

Appendix F explains the standard collating sequence used to sort items of data into a particular order.

### **Appendix G. Understanding Railroad Diagrams**

Appendix G explains how to read railroad diagrams.

## Related Product Information

***A Series CANDE Operations Reference Manual (form 8600 1500)***

This manual describes how CANDE operates to allow generalized file preparation and updating in an interactive, terminal-oriented environment. This manual is written for a wide range of computer users who work with text and program files.

***A Series Editor Operations Guide (form 8600 0551)***

This guide describes the operation of the Editor, an interactive tool for creating and modifying text and program files. This guide is written for experienced and inexperienced users who are responsible for creating and maintaining text and program files.

***A Series File Attributes Programming Reference Manual (form 8600 0064).  
Formerly A Series I/O Subsystem Programming Reference Manual.***

This manual contains information about each file attribute and each direct I/O buffer attribute. The manual is written for programmers and operations personnel who need to understand the functionality of a given attribute.

***A Series MultiLingual System (MLS) Administration, Operations, and  
Programming Guide (form 8600 0288)***

This guide describes how to use the MLS environment, which encompasses many Unisys products. The MLS environment includes a collection of operating system features, productivity tools, utilities, and compiler extensions. The guide explains how these products are used to create application systems tailored to meet the needs of users in a multilingual or multicultural business environment. It explains, for example, the procedures for translating system and application output messages, help text, and user interface screens from one natural language to one or more other languages; for instance, from English to French and Spanish. This guide is written for international vendors, branch systems personnel, system managers, programmers, and customers who wish to create customized application systems.

***A Series System Software Installation Guide, Volume 2: System Initialization  
(form 1170263)***

This guide provides the various step-by-step system initialization procedures for most A Series Entry and Medium Systems. This guide is written for installation managers, support analysts, and operators.

***A Series System Software Support Reference Manual (form 8600 0478)***

This manual describes a number of facilities used for system monitoring and debugging, including BARS, DUMPANALYZER, LOGANALYZER, and LOGGER. It also describes the format of the SUMLOG file. This manual is written for system support personnel and operators.

***A Series System Software Utilities Operations Reference Manual  
(form 8600 0460)***

This manual provides information on the system utilities, such as DCSTATUS, FILECOPY, and DUMPALL. This manual is written for applications programmers and operators.

***A Series Work Flow Language (WFL) Programming Reference Manual (form  
8600 1047)***

This manual presents the complete syntax and semantics of WFL. WFL is used to construct jobs that compile or run programs written in other languages and that perform library maintenance such as copying files. This manual is written for individuals who have some experience with programming in a block-structured language such as ALGOL and who know how to create and edit files using CANDE or the Editor.



# Contents

	About This Manual .....	v
<b>Section 1.</b>	<b>Creating and Compiling Sort Programs</b>	
	Creating Sort Programs .....	1-1
	Compiling and Running Sort Programs .....	1-2
	Compiling and Running Sort Programs from CANDE ..	1-3
	Compiling and Running Sort Programs from WFL .....	1-4
	Compiler Control Options .....	1-5
	Using the TASKVALUE Task Attribute .....	1-6
<b>Section 2.</b>	<b>Sort Program Structure</b>	
	Overall Sort Program Structure .....	2-1
	Sort Program Record Format .....	2-2
	Comments .....	2-2
	Basic Syntax Components .....	2-3
	Integer .....	2-3
	Displacement .....	2-3
	Type .....	2-5
	Length .....	2-9
	Literal .....	2-9
	Literal Conversions .....	2-11
	Literals in DIGIT and ZONE Fields .....	2-11
	Hex Literal Conversion .....	2-12
	Integer Literal Conversion .....	2-12
<b>Section 3.</b>	<b>SORT Statements</b>	
	FILE Statement .....	3-1
	File Attributes .....	3-2
	KEY Statement .....	3-3
<b>Section 4.</b>	<b>Optional SORT Statements</b>	
	SORT Processing Statements .....	4-1
	COLLATE Statement .....	4-1
	DISKANDTAPE Statement .....	4-2
	DISKSORT Statement .....	4-2
	MEMORYONLY Statement .....	4-3
	MERGE Statement .....	4-3
	STABLE Statement .....	4-4
	TAGSORT Statement .....	4-5



TAPESORT Statement .....	4-6
<b>SORT Parameter Statements</b> .....	4-7
MEMORY Statement .....	4-7
PARITY DISCARD Statement .....	4-8
RECORDS Statement .....	4-9
TAGSEARCH Statement .....	4-9
WORKFAMILY Statement .....	4-10
WORKSIZE Statement .....	4-10
<b>INCLUDE and DELETE Record Selection Statements</b> .....	4-11

**Section 5. SORT Compiler Control Records**

<b>Types of Compiler Control Records</b> .....	5-1
<b>Types of Options</b> .....	5-1
<b>Using TASKSTRING to Submit Compiler Control Records</b> .....	5-2
<b>Compiler Control Record Syntax</b> .....	5-2
<b>Compiler Control Options</b> .....	5-4
CLEAR (Immediate) .....	5-4
CODE (Boolean) .....	5-4
DELETE (Boolean) .....	5-4
ERRORLIMIT (Value) .....	5-5
ERRORLIST (Boolean) .....	5-6
GSORT (Boolean) .....	5-6
LIST (Boolean) .....	5-7
LISTDELETED (Boolean) .....	5-7
LISTDOLLAR (Boolean) .....	5-8
LISTP (Boolean) .....	5-8
MAP (Boolean) .....	5-8
MERGE (Boolean) .....	5-9
NEW (Boolean) .....	5-9
NEWSEQERR (Boolean) .....	5-10
REFORMAT (Boolean) .....	5-11
SEQUENCE (Boolean) .....	5-11
Sequence Base (Value) .....	5-12
Sequence Increment (Value) .....	5-12
SINGLE (Boolean) .....	5-12
VERSION (Value) .....	5-12

**Appendix A. SORT Language Keywords**

**Appendix B. SORT Compiler Files**

**Appendix C. SORT Program Files**

**Appendix D. B 1000 SORT Conversion**

<b>B 1000 Statements Accepted but Ignored</b> .....	D-1
<b>B 1000 Statements Not Supported by the SORT Compiler</b> .....	D-1

<b>B 1000 File Statement</b> .....	D-1
B 1000 File Input Part .....	D-2
B 1000 File Output Part .....	D-4
B 1000 File Name .....	D-5
<b>B 1000 Data Type</b> .....	D-7
<b>B 1000 INCLUDE and DELETE Statement</b> .....	D-9
<b>B 1000 MEMORY Statement</b> .....	D-9
<b>B 1000 MERGE Statement</b> .....	D-9
<b>B 1000 NOPRINT Statement</b> .....	D-9
<b>B 1000 RECORDS Statement</b> .....	D-10
<b>B 1000 SORT Statement</b> .....	D-10
<b>B 1000 SYNTAX Statement</b> .....	D-10
<b>B 1000 TAGSORT Statement</b> .....	D-10
<b>B 1000 TAPESORT Statement</b> .....	D-11
<b>B 1000 WORKPACK1 and WORKPACK2 Statements</b> .....	D-11

**Appendix E. Using GSORT to Code and Run Sort Programs**

<b>How GSORT Programs Are Similar to Sort Programs</b> .....	E-1
<b>How GSORT Programs Differ from Sort Programs</b> .....	E-2
Creating Output Records by Remapping Input Records .....	E-2
Using Alternate Collating Sequences .....	E-3
Coding GSORT Programs .....	E-4
Executing a GSORT Program .....	E-5
<b>Determining the Content of the Output File</b> .....	E-6
Criteria for Regularly Sorted Output (SORTR) .....	E-6
Criteria for Sorted Files with Accumulated Totals (SORTRS) .....	E-7
Criteria for Files of Record Numbers (SORTA) .....	E-7
Selecting the Fields in an Output Record .....	E-8
<b>Coding Header Information</b> .....	E-8
Sequence Number (Columns 1 through 5) .....	E-10
Header Specification (Column 6) .....	E-10
Type of Job (Columns 7 through 12) .....	E-11
Identical, or Equal, Key Fields (Column 12) .....	E-11
Key Field Lengths (Columns 13 through 17) .....	E-11
Record-Sorting Order (Column 18) .....	E-12
Columns 19 through 25 .....	E-12
Collating Sequence (Column 26) .....	E-12
Print Option (Column 27) .....	E-13
Output Option (Column 28) .....	E-13
Output Record Length (Columns 29 through 32) .....	E-13
Columns 33 through 35 .....	E-14
Null Output (Column 36) .....	E-14
Columns 37 through 43 .....	E-14
Comments (Column 44 through 72) .....	E-14
<b>Coding ALTSEQ Statements</b> .....	E-14
Name of Statement (Columns 1 through 6) .....	E-15
Double Asterisks (Columns 1 and 2) .....	E-15
(Columns 7 and 8) .....	E-16
Character Being Taken Out (Columns 9 and 10) .....	E-16
Character Being Inserted (Columns 11 and 12) .....	E-16

## Contents

---

Using Alternate Collating Sequences for an Entire Key Field .....	E-16
Using Alternate Collating Sequences for Specified Key Fields .....	E-17
Using the ALTSEQ Statement to Make Characters Equal	E-17
<b>Coding Record Selection Information</b> .....	E-18
Sequence Number (Columns 1 through 5) .....	E-20
Include/Omit Specifications (Column 6) .....	E-20
Using the Include-All Option .....	E-20
Using the Conditional Include Option .....	E-20
Mixing the Include and Omit Options .....	E-20
Continuation or Comments (Column 7) .....	E-21
Interpretation of Data (Column 8) .....	E-25
Factor 1 Location (Columns 9 through 16) .....	E-26
Relation (Columns 17 and 18) .....	E-27
Factor 2 Compare Data (Column 19) .....	E-28
Factor 2 Location (Columns 20 through 27) .....	E-28
Compare Field Name (Columns 28 through 39) .....	E-28
Compare Literal or Constant (Columns 20 through 39) .....	E-29
Compare Date Keyword (Columns 20 through 39) .....	E-30
Comments (Column 40 through 72) .....	E-30
<b>Coding Field Selection Information</b> .....	E-30
Sequence Number (Columns 1 through 5) .....	E-32
Field Selection Specifications (Column 6) .....	E-32
Field Type or Comments (Column 7) .....	E-33
Defining Types of Key Fields .....	E-33
Normal Key Fields .....	E-34
Opposite Key Fields .....	E-34
Forced Key Fields .....	E-34
Unconditional Forced Field .....	E-34
Conditional Forced Field .....	E-34
Force-All Field .....	E-35
Specifying Types of Data Fields .....	E-35
Normal Data Fields .....	E-35
Summary Data Fields .....	E-35
Forced Data Fields .....	E-36
Data Type (Column 8) .....	E-36
Field Location (Columns 9 through 16) .....	E-38
Compare Character (Column 17) .....	E-39
For a Forced Key Field .....	E-39
For a Summary Overflow Indicator Field .....	E-39
Forced Character (Column 18) .....	E-39
Continued Forced Character in Forced Key Field (Column 19) .....	E-40
Overflow Field Length (Columns 20 through 22) .....	E-40
Preventing Summary Data Overflow .....	E-41
Detecting Summary Data Overflow .....	E-41
Alternate Collating Sequences (Column 20) .....	E-41
Columns 23 through 39 .....	E-41
Comments (Columns 40 through 72) .....	E-42
<b>Examples of GSORT Programs</b> .....	E-42
Sorting without Record Selection Specifications .....	E-42

Sorting a File with Selected Records .....	E-43
Sorting a File with Conditional Include and Omit .....	E-44
Sorting a File by Record Number .....	E-47
Sorting for Selected Information .....	E-48
Sorting by Integer .....	E-49
<b>GSORT System Error Messages .....</b>	<b>E-50</b>

**Appendix F. Using the Standard Collating Sequence**

Comparing Both the Zone and Digit Portions .....	F-1
Comparing Only the Zone Portion .....	F-3
Comparing Only the Digit Portion .....	F-7

**Appendix G. Understanding Railroad Diagrams**

<b>Glossary .....</b>	<b>1</b>
<b>Bibliography .....</b>	<b>1</b>
<b>Index .....</b>	<b>1</b>



# Figures

G-1.	Railroad Constraints .....	G-5
------	----------------------------	-----



# Tables

1-1.	TASKVALUE Task Attribute Values . . . . .	1-6
2-1.	Key and Data Field Storage . . . . .	2-8
2-2.	Allowable Literal Types . . . . .	2-12
A-1.	Category 1 Keywords . . . . .	A-1
A-2.	Category 2 Keywords . . . . .	A-2
B-1.	SORT Compiler Files . . . . .	B-1
C-1.	Sort Program Files . . . . .	C-1
D-1.	Equating B 1000 File Names and A Series TITLE Attributes . . . . .	D-6
E-1.	Header Specification Entry Table . . . . .	E-9
E-2.	ALTSEQ Statement Entry Table . . . . .	E-15
E-3.	Record Selection Specification Entry Table . . . . .	E-18
E-4.	Coding for Include Sets . . . . .	E-21
E-5.	Coding for Omit Sets . . . . .	E-24
E-6.	Coding for Negative Unpacked Numbers . . . . .	E-30
E-7.	Field Selection Specification Entry Table . . . . .	E-32
E-8.	Possible Entry Combinations (Columns 7 and 8) . . . . .	E-38
E-9.	Maximum Field Lengths for Input Data Types . . . . .	E-41
F-1.	EBCDIC Collating Sequence for Comparing Both the Zone and Digit Portions of a Character . . . . .	F-2
F-2.	EBCDIC Collating Sequence for Comparing Only the Zone Portion of a Character . . . . .	F-4
F-3.	EBCDIC Collating Sequence for Comparing Only the Digit Portion of a Character . . . . .	F-8





# Section 1

## Creating and Compiling Sort Programs

The SORT language allows users to write programs for sorting or merging files on Unisys A Series systems. A single Sort program can sort up to 99 files or merge from two to eight files. Files can be sorted in main memory, on tape, on disk, or on tape and disk, depending on the type of sort specified in the sort program. However, files are always merged in main memory, because merges require much less memory than sorts.

The SORT language (and associated compiler) simplifies the use of the Master Control Program (MCP) SORT procedure because the procedure no longer has to be accessed through other programming languages, such as ALGOL or COBOL.

Sort programs can specify one or many key fields (up to 200) for processing input data records. The key field or fields can begin at any location within the input record, and can be of any specified length not exceeding the length of the record. Individual keys can be sorted or merged in ascending or descending order. Specific records can be included or excluded from the sort or merge process through optional sort selection expressions.

Sort programs can be written using the Command and Edit (CANDE) Message Control System (MCS) or the Editor. They can be compiled and run from either CANDE or the Work Flow Language (WFL).

The SORT compiler supports the MultiLingual System (MLS). Through the Message Translation utility, a component of the MLS, you can translate all error, warning, header, and trailer messages issued by the compiler.

This section explains how to create and compile Sort programs.

### Creating Sort Programs

Sort programs can be created and modified by using CANDE directly or by using the Editor, which is accessed through CANDE.

Sort program files can be created, modified, and saved using the CANDE *MAKE*, *GET*, and *SAVE* commands. The *MAKE* command creates a new file. The *GET* command retrieves a previously created and saved file from the user's program library. The *SAVE* command saves a new or modified file in the user's program library.

When writing Sort programs via CANDE, the programmer must enter statement numbers. When the Editor is used, statement numbering is done automatically.

**Note:** *Sort programs must be written using uppercase characters only. The SORT compiler will generate a syntax error (INVALID CHARACTER) if a lowercase character is encountered in the program file.*

For more information on creating, modifying, and saving files using CANDE, refer to the *CANDE Operations Reference Manual*. For information on using the Editor, refer to the *Editor Operations Guide*.

### **Example: Creating and Saving a Sort Program Using CANDE**

This example shows how to create and save a Sort program file using CANDE. Lines beginning with a number sign (#) are displayed by CANDE; all other lines are entered by the user.

```
MAKE SRTPROG SORT

#WORKFILE SRTPROG: SORT

100 DISKSORT
200 FILE IN (TITLE ="INPUT/FILE")
300 FILE OUT (TITLE ="OUTPUT/FILE")
400 KEY (1 5 D)
SAVE

#UPDATING
#WORKSOURCE SRTPROG SAVED
```

In the MAKE command line, the keyword SORT, specifies that the file SRTPROG is to be of type SORT. (Sort program files can also be created as type SEQUENTIAL or type TEXT. However, doing so changes the form of the COMPILE command. These differences are described in "Compiling and Running Sort Programs from CANDE" later in this section.)

The program code, written in all uppercase, designates that a disk sort is to be performed on the input file INPUT/FILE and the sorted output should be placed in the file OUTPUT/FILE. By default, the key field is of type ALPHA. The key field has a stated displacement of 1 and a length of 5. The key should be sorted in descending order.

The SAVE command saves the new file SRTPROG.

## Compiling and Running Sort Programs

The SORT compiler can be initiated from either the Work Flow Language (WFL) or CANDE, and the object code files generated by the SORT compiler can be run from either CANDE or WFL.

Programs compiled by the SORT compiler use the system SORT procedure in the Master Control Program (MCP). For more information about the capabilities and mechanism of the procedure, refer to the *System Software Utilities Operations Reference Manual*.

## Compiling and Running Sort Programs from CANDE

When accessed from CANDE, the SORT compiler can compile programs either for syntax only (the COMPILE SYNTAX command) or for storage in the program library and immediate execution (the COMPILE command). Additionally, if a Sort program file is the current CANDE work file, it can be both compiled and run using only the CANDE RUN command.

If a Sort program file has been created as a SEQUENTIAL or TEXT file, then the COMPILE command must explicitly state which compiler is to be used (in this case it is the SORT compiler). The COMPILE command would then be either of the following:

```
COMPILE WITH SORT

COMPILE SRTPROG WITH SORT
```

If a syntax error is diagnosed during the compilation of a Sort program, the compilation is halted and no object code file is produced. The compiler sends messages to the screen indicating the lines that contain errors, along with an explanation of each error.

For more information on compiling and running files using CANDE commands, refer to the *CANDE Operations Reference Manual*.

### Example: Compiling with CANDE for Syntax Only

The following example shows how to compile, for syntax only, a Sort program file named SRTPROG that has been saved in the user's file library. The compilation does not create an object code file.

```
COMPILE SRTPROG SYNTAX

#COMPILING 2345
#ET=8.2 PT=0.4 IO=0.5
```

SRTPROG contains no syntax errors. It can now be compiled and run using either of the following command sequences. Both compilations produce an object code file.

Combined Compile and Run	Separate Compile and Run
GET SRTPROG	GET SRTPROG
#WORKFILE SRTPROG:SORT, 4 RECORDS, SAVED	#WORKFILE SRTPROG:SORT, 4 RECORDS, SAVED
RUN	COMPILE
#COMPILING 1234	#COMPILING 2345
#ET=4.0 PT=0.4 IO=0.8	#ET=3.6 PT=0.4 IO=0.7
#RUNNING 5678	
#ET=1.5 PT=0.3 IO=0.5	SAVE
SAVE	#WORKOBJECT SRTPROG SAVED
#WORKOBJECT SRTPROG SAVED	

### Combined Compile and Run

### Separate Compile and Run

RUN

#RUNNING 3456

#ET=2.9 PT=0.5 IO=0.6

### Example: Using File Attributes in a CANDE Job

This example shows how to run the previously compiled and saved Sort program, SRTPROG, using optional file attribute equations to specify different input and output files to be used by the program. Notice that in a file attribute equation the title of the file does not need to be enclosed by double quotation marks (").

```
RUN SRTPROG; FILE IN(TITLE=NEW/INPUT); FILE OUT(TITLE=NEW/OUTPUT)
```

```
#RUNNING 9012
```

```
#ET=5:04.3 PT=4:21.3 IO=2:48.1
```

## Compiling and Running Sort Programs from WFL

When accessed through WFL, depending on what code file job disposition is specified, the SORT compiler can compile programs for syntax only (SYNTAX), for storage in the program library (LIBRARY), for immediate execution (GO), or for storage in the library and immediate execution (LIBRARY GO).

The following examples show only some of the basic constructs for compiling and running Sort programs from WFL. For complete information about the WFL *COMPILE* and *RUN* commands, refer to the *Work Flow Language (WFL) Programming Reference Manual*.

### Example: WFL Compilation for Storage in a Library

In the following example, the Sort program SORTPROG is compiled under the WFL job COMPILE/SORT. The Sort program is contained in a Local Data Specification (the lines beginning with SORT DATA CARD and ending with the question mark), and the object code is to be written to a file named SORTPROG. The keyword SORT on the second line is the call to the SORT compiler. The keyword LIBRARY specifies that SORTPROG is to be stored in the user's file library.

Once compiled, SORTPROG can be run at any time. The question marks (?) preceding the BEGIN JOB and END JOB statements, and the question mark following the KEY statement (it closes the Local Data Specification) must be placed in column 1.

```
? BEGIN JOB COMPILE/SORT;
  COMPILE SORTPROG WITH SORT LIBRARY;
  SORT DATA CARD
    FILE IN (TITLE = "INPUT/FILE")
    FILE OUT (TITLE = "OUTPUT/FILE")
    KEY (1 5)
? % END OF SORT COMPILER DATA
? END JOB
```

**Example: WFL Compilation for Immediate Execution**

This example shows a compilation for immediate execution of a Sort program. The Sort program ALPHASORT had been previously written and saved in the user's file library. The object code file is to be named OBJ/ALPHASORT.

```
? BEGIN JOB COMPILE/SORT;
  COMPILE OBJ/ALPHASORT WITH SORT GO;
  COMPILER FILE CARD (TITLE="ALPHASORT");
? END JOB
```

**Example: Use of File Attributes in WFL Jobs**

Below are two different runs of the Sort program SORTPROG, compiled in the example "WFL Compilation for Storage in a Library." The first run uses the file attributes specified when the program was compiled; the second run respecifies the file attribute, TITLE, for FILE OUT to be NEW/OUTPUT/FILE rather than OUTPUT/FILE. Comments can be added to the JOB file by preceding them with a percent symbol (%).

```
? BEGIN JOB RUN/SORT;
  RUN SORTPROG; % sorts INPUT/FILE, producing OUTPUT/FILE

  RUN SORTPROG; % sorts INPUT/FILE, producing NEW/OUTPUT/FILE
  FILE OUT (TITLE = NEW/OUTPUT/FILE);
? END JOB
```

**Compiler Control Options**

The SORT compiler accepts the following compiler control options:

CLEAR	CODE	DELETE
ERRORLIMIT	ERRORLIST	GSORT
LIST	LISTDELETED	LISTP
LISTDOLLAR	MAP	MERGE
NEW	NEWSEQERR	REFORMAT
SEQUENCE	SINGLE	VERSION

For complete information about the compiler control options listed above, refer to the "Compiler Control Records" section in this manual.

## Using the TASKVALUE Task Attribute

Sort programs use the TASKVALUE task attribute to return information about program execution. Table 1-1 shows the values that can be returned.

**Table 1-1. TASKVALUE Task Attribute Values**

Value	Meaning
1	The Sort program terminated normally.
0	Either the Sort program is not yet initiated, or the Sort program is still running.
-1	The Sort program terminated because of a run-time error condition.
-2	The Sort program was discontinued for some external reason (for example, by the system operator).

The TASKVALUE of a Sort program can be tested in a WFL job that uses a task variable. The following example shows how the TASKVALUE can be used to provide Sort program information.

### **Example: Providing Sort Program Information with TASKVALUE**

In the example, the WFL job RUN/SORT can produce one of two messages. If the program SORTPROG terminates normally, the message "SORT RAN OK" will be displayed. Otherwise, the message "SORT DID NOT WORK" is displayed.

```
? BEGIN JOB RUN/SORT;
  TASK T;
  RUN SORTPROG[T];
  IF T(TASKVALUE) = 1 THEN
    DISPLAY "SORT RAN OK"
  ELSE
    ABORT "SORT DID NOT WORK";
? END JOB
```

# Section 2

## Sort Program Structure

This section describes the structure of a Sort program and the basic syntax components of SORT statements.

### Overall Sort Program Structure

A Sort program is composed of a series of statements that define both the files to be sorted or merged and the data fields to be used as keys within the records of those files.

Shown below are the valid SORT statements. The statements specify the actions of the Sort program. A Sort program must have one FILE statement and one KEY statement. A Sort program can be written using only these required statements. However, most Sort programs contain optional statements that refine the sort or improve sorting efficiency. Note that the statements do not have to appear in any specific order.

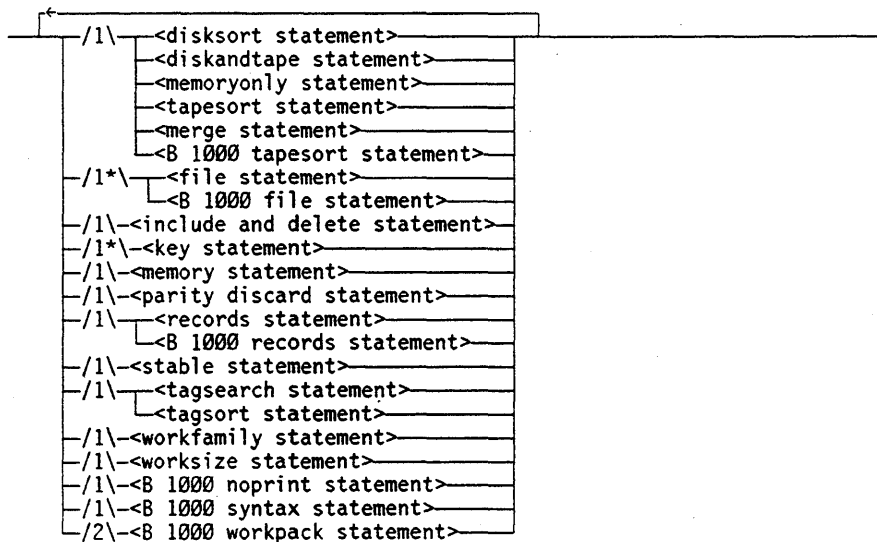
For detailed information about individual statements, refer to the "SORT Statements" and "Optional SORT Statements" sections in this guide. For information describing the compatibility of SORT statements with the B 1000 Sort program, refer to the "B 1000 SORT Conversion" appendix.

#### Syntax

<sort program>

—<sort statements>—

<sort statements>





Below is a basic outline for a Sort program. Following the outline makes the program code easier to read.

Sort Processing Statements	-- optional
File Statement	-- REQUIRED
Input files (up to 99)	
Output file (one only)	
Record Selection Statement	-- optional
Include/Delete statement (up to 200)	
Key Statement	-- REQUIRED
Key field (up to 200)	
Sort Parameter Statements	-- optional

Compiler control records (CCRs) are not shown in the outline because their locations within a Sort program depend on what the programmer is trying to control.

*Note: GSORT programs do not use the SORT statements; they use the GSORT statement and specifications. These elements must be entered in a specific order. See Appendix E, "Using GSORT to Code and Run Sort Programs."*

## Sort Program Record Format

Sort program input records must contain SORT language statements in columns 1 through 72. Optional sequence numbers can appear in columns 73 through 80. The sequence number field is used in conjunction with the \$MERGE and \$SEQUENCE compiler control options. Optional version (or patch) information can appear in columns 81 through 90. The version field is used by the \$VERSION compiler control option; however, if the \$VERSION option is not used, then the version field is not used by the SORT compiler and can contain any textual information.

More than one SORT statement can appear on a single input record; however, each statement must be separated from the next by a semicolon (;).

Sort keywords, integers, and literals cannot be split across an input record boundary.

*Note: Program input records for GSORT programs have a different format and different rules. Appendix E, "Using GSORT to Code and Run Sort Programs," details the GSORT program input record.*

## Comments

Comments can be included in a Sort program. The comments are preceded by a percent sign (%) or a colon (:). When either symbol is encountered by the SORT compiler, scanning is terminated on that input record. However, if the percent sign or colon appears within a quoted string, it is not treated as a comment indicator.

*Note: Comments are column-dependent in GSORT programs. See Appendix E, "Using GSORT to Code and Run Sort Programs," for information.*

## Basic Syntax Components

Certain syntactical elements can be termed basic because of their repeated occurrence in SORT statements. These elements are discussed in detail now to reduce repetition. Later sections of the manual will refer the reader to this section instead of repeating syntax diagrams and detailed explanations.

### Integer

Integers are contained in many SORT statements. An `<integer>` is a whole number value having a maximum of 11 digits.

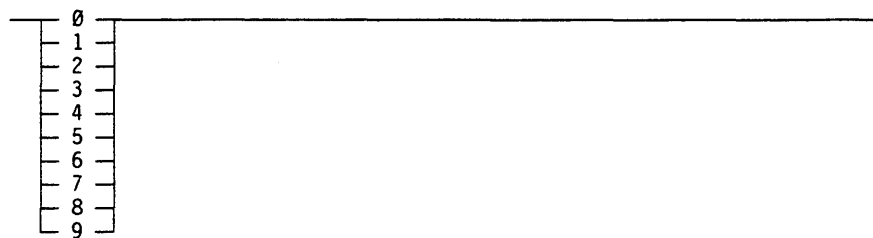
An `<integer>` should not be confused with an `<integer literal>`. An `<integer literal>` is a data item used for comparison with key fields, or for storage into data fields within the records that are being sorted or merged.

#### Syntax

`<integer>`



`<digit>`



#### Explanation

`<integer>`

Contains 1 to 11 digits and cannot be split across an input record boundary.

`<digit>`

Any character in the range of 0 through 9, inclusive.

### Displacement

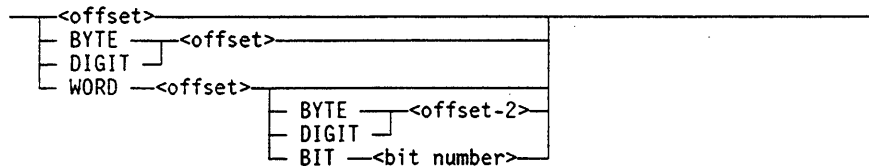
A displacement specifies the position of the first, or most significant, element in a key or data field. The first unit of a record is considered to be relative position 1. Unless explicitly specified, the position of a data or key field within a record is determined with the default units that apply to the `<type>` of the key or data field. The default units are bytes for all types except PACKED and HEX, which have 4-bit default units. If the

type of the field includes a LEADING SEPARATE sign, the displacement points to this sign character.

If a specified displacement is greater than the record size of any record to be sorted, a run-time error results.

### Syntax

<displacement>



<offset>

—<integer>

<offset-2>

—<integer>

### Explanation

< offset >, < offset-2 >

Specifies the position of the first, or most significant, element in the key or data field.

BYTE < offset >

Specifies bytes as the displacement units. The offset value is the number of bytes to displace from position 1. BYTE is the default displacement for all data types except PACKED, HEX, and BIT.

DIGIT < offset >

Specifies 4-bit digits as the displacement units. The offset value is the number of 4-bit units to displace from position 1. DIGIT is the default displacement for PACKED and HEX data types.

WORD < offset >

Specifies 48-bit words as the displacement units. The offset value is the number of 48-bit units to displace from position 1. If a WORD displacement is specified, a further displacement specification of BYTE, DIGIT, or BIT can be declared. For example, WORD 2 BYTE 3 specifies a displacement to the third byte within the second word of the record.

BIT < bit number >

Specifies single bit displacement units. A BIT-displacement can be used only when BIT is the specified type. When specifying a BIT-displacement, WORD < offset > must be

declared first, followed by BIT < bit number > . The leading, or leftmost, bit in a word is numbered bit 47. The trailing, or rightmost, bit in a word is numbered bit 0. For example, WORD 3 BIT 47 specifies that the key field is to begin at the leftmost bit of the third word of each record. BIT is the default displacement for the BIT data type.

< bit number >

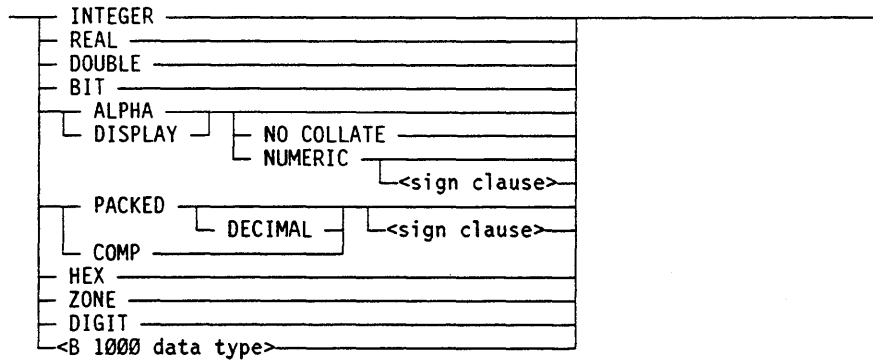
Specifies an integer within the range 0 through 47, inclusive.

## Type

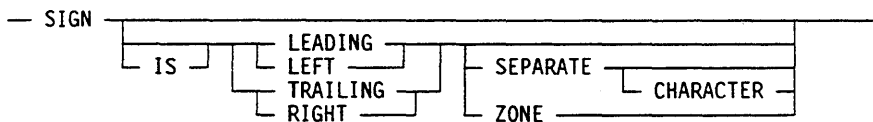
A type defines the format of a data item. ALPHA is the default type.

### Syntax

<type>



<sign clause>



### Explanation

#### INTEGER

Specifies that the key or data field is a 48-bit value, maintained with an exponent of 0 and no fractional part. A < length > of 1 must be specified in the < key field >; otherwise, a syntax error results. The default displacement units are bytes.

#### REAL

Specifies that the key or data field is a 48-bit value that can have an exponent and a fractional part. A < length > of 1 must be specified in the < key field >; otherwise, a syntax error results. The default displacement units are bytes.

### DOUBLE

Specifies a 96-bit entity, internally carried as two 48-bit words, for the key or data field. A <length> of 1 must be specified in the <key field>; otherwise, a syntax error results. The default displacement units are bytes.

### BIT

Specifies the key or data field to be a sequence of bits. The bit sequence size is specified by the <length> parameter, which must be a value in the range 1 through 48, inclusive.

### ALPHA, DISPLAY

Specifies 8-bit EBCDIC characters for the key or data field. ALPHA is the default type for key fields in Sort programs. When the key field type is ALPHA, a displacement is not assumed by default; the BIT displacement must be explicitly specified.

### NO COLLATE

Used for program documentation. NO COLLATE is ignored by the SORT compiler.

### ALPHA NUMERIC, DISPLAY NUMERIC

Specifies 8-bit numeric units for the key or data field. Only the digit portion, which is the low-order four bits of each 8-bit character, is used to determine the numeric value of the field. The zone portion, which is the high-order four bits, is ignored. The digit portion of each character can contain values only in the range 0 through 9. The values A through F are not allowed. The displacement units are bytes.

### < sign clause >

Indicates the presence, location, and type of a sign for a numeric quantity (positive or negative). If a sign is present, the key field is evaluated based on the algebraic value of the digits in the field after the sign has been applied.

### PACKED, COMP

Specifies 4-bit numeric units. Only the digit values 0 through 9 are allowed when the numeric value of a PACKED data or key item is computed. The values A through F are not acceptable. If the PACKED key field is unsigned, the sort comparison is performed in such a way that it allows the full range of hexadecimal values to be used in the key field. (This comparison is done to ensure B 1000 data compatibility.) For PACKED SIGN key or data fields, the hex digits A to F cannot appear in the digit portion of the field. The value D in the sign position represents the negative sign (-); any other value found in the sign position represents the positive sign (+). The sign of a PACKED field is always in a separate character; therefore, the keyword SEPARATE is ignored by the SORT compiler if used in a PACKED field designation. The displacement units are 4-bit digits for PACKED fields. (Refer to "Displacement" in this section.)

### DECIMAL

Used for program documentation. DECIMAL is ignored by the SORT compiler.

### HEX

Specifies 4-bit, hexadecimal characters for the key or data field. The default displacement is 4-bit units, and all hexadecimal characters (0 through 9 and A through F) are allowed.

### ZONE

Specifies bytes for the key or data field. Only the zone portion, which is the high-order four bits of each byte, determines the value of the field. The default displacement units are bytes.

### DIGIT

Specifies bytes for the key or data field. Only the digit portion, which is the low-order four bits of each byte, is used during sorting. The default displacement units are bytes. A DIGIT type is equivalent to an unsigned ALPHA NUMERIC option in a key statement, but has a slightly different meaning if it is used with a literal. (Refer to "Literal" in this section.)

### < B 1000 data type >

Specifies that a B 1000 Sort program data representation will be used and that the compiler should accept it. For information about B 1000 data types and compatibility refer to the "B 1000 SORT Conversion" appendix.

### IS, IS CHARACTER

Used for program documentation. Both the IS and the IS CHARACTER are ignored by the SORT compiler.

### TRAILING ZONE, RIGHT ZONE

For ALPHA data types, specifies that the sign is in the digit portion of the last character in the field. For PACKED or COMP data types, this option specifies that the sign is in the rightmost four bits of the field. A hexadecimal D in this position signifies a negative sign (-). Any other value in this position represents a positive sign (+). TRAILING is the default location of the sign character for ALPHA NUMERIC fields when only SIGN is specified.

The keyword ZONE can be omitted. For example, TRAILING ZONE and TRAILING have the same meaning.

### LEADING ZONE, LEFT ZONE

For ALPHA data types, specifies that the sign is in the zone portion of the first character in the field. For PACKED or COMP data types, this option specifies that the sign is in the leftmost four bits of the field. A hexadecimal D in this position signifies a negative

sign (-). Any other value in this position represents a positive sign (+). LEADING is the default location of the sign character for PACKED fields when only SIGN is specified.

The keyword ZONE can be omitted. For example, LEADING ZONE and LEADING have the same meaning.

### LEADING SEPARATE, LEFT SEPARATE

Specifies that the sign is a separate 8-bit character immediately preceding the first numeric character in the field. The displacement must point to the sign, and the length must include the sign character. (Refer to "Displacement" in this section.) The negative sign is the EBCDIC minus sign (-). Any other value is treated as a positive sign (+). For PACKED or COMP data types, SEPARATE is used for program documentation and is ignored by the SORT compiler. The sign of a PACKED field is always a separate character.

### TRAILING SEPARATE, RIGHT SEPARATE

Specifies that the sign is a separate 8-bit character immediately following the last numeric character in the field. The length must include the sign character. The negative sign is the EBCDIC minus sign (-). Any other value is treated as a positive sign (+). For PACKED or COMP data types, SEPARATE is used for documentation purposes only. A PACKED field sign is always a separate character.

Table 2-1 lists how each data type is represented and stored internally, and gives the method of comparing key fields with record fields for each data type.

**Table 2-1. Key and Data Field Storage**

Data Type	Item Storage	Method of Comparison
INTEGER	1 word, 48 bits byte-aligned	Full arithmetic.
REAL	1 word, 48 bits byte-aligned	Full arithmetic.
DOUBLE	2 words, 96 bits byte-aligned	Full arithmetic.
BIT	n bits, n <= 48, 1 word only	n-bit binary.
ALPHA	n bytes, 8 bits byte-aligned	Full 8-bit.
ALPHA NUMERIC	n bytes, 8 bits byte-aligned with optional sign	Numeric value of digits. Hex digit values A through F are illegal. Sign either separate or zone.
PACKED SIGNED	n digits, 4 bits digit-aligned	Numeric value of digits. Hex digit values A through F are illegal. Separate sign.
PACKED (unsigned)	n digits, 4 bits digit-aligned	Full 4-bit.

continued

Table 2-1. Key and Data Field Storage (cont.)

Data Type	Item Storage	Method of Comparison
HEX	n digits, 4 bits digit-aligned	Full 4-bit.
ZONE	n bytes, 8 bits byte-aligned	Hex value of zone bits.
DIGIT	n bytes, 8 bits byte-aligned	Hex value of digit bits.

## Length

The length specifies the number of units in the key or data field and includes the sign character if the < sign clause > is specified with the field type. The length plus the displacement of the field cannot exceed the record size of any record being sorted; otherwise, a run-time error occurs.

### Syntax

<length>

—<integer>—

## Literal

A literal is a character string whose value is specified by an ordered set of characters. This character string is a word, number, or symbol that names, describes, or defines itself and not something else that it might represent. A literal cannot cross an input record boundary.

### Syntax

<literal>

—<graphic literal>—  
 —<hex literal>—  
 —<integer literal>—  
 —<real literal>—



## Sort Program Structure

---

<graphic literal>

— " /70\-<graphic> "

<graphic>

— <Any EBCDIC character except double quotes>

<hex literal>

— @ /70\-<hex digit> @

<hex digit>

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

<integer literal>

— # /11\-<digit> #

+  
-  
.

<real literal>

— # /24\-<digit> /1\-. E D + /5\-<digit>

→ #

### Explanation

< graphic literal >

Specifies a literal composed of EBCDIC characters that is left-justified and padded with blanks on the right.

< hex literal >

Specifies a literal composed of hex digits (numbers 0 through 9 and alphabetic characters A through F). The literal is right-justified and, when necessary, padded with hexadecimal zeros on the left.

< integer literal >

Specifies a literal that is based on an integer arithmetic value. The associated length must have the value of one word, which is 6 bytes or 12 digits.

< real literal >

Specifies a literal that is based on a real arithmetic value. The associated length must have the value of 1, which represents one single-precision word (6 bytes or 12 digits) or one double-precision word (12 bytes or 24 digits). Also, a real literal allows a full floating-point literal to be used as a data item.

E

Specifies that the < real literal > is a single-precision word.

D

Specifies that the < real literal > is a double-precision word.

< digit >

Specifies the power of 10 by which the number is to be multiplied when used after E or D in a < real literal >. A digit is syntactically defined under "Integer" in this section.

## Literal Conversions

Literals can be converted to other literal types based on the type of field with which the literal is compared. This feature allows programmers to express literals in the most convenient form. Table 2-2, later in this section, lists the fields that can be used with each literal type.

## Literals in DIGIT and ZONE Fields

When a literal is associated with a field of type ZONE or DIGIT, the actual length of the literal must be exactly one graphic character or two hex characters; otherwise, a syntax error occurs. When using these types, the SORT compiler builds a literal value of the specified length by using either the zone or digit bits from the specified literal.

For type ZONE, each byte in the constructed literal is first filled with the bit configuration 01000000 (hex 40) for the specified length, and then the zone portion of the literal is moved to the zone portion of each character in the field.

For type DIGIT, each byte in the constructed literal is first filled with the bit configuration 11110000 (hex F0) for the specified length, and then the digit portion of the specified literal is moved to the digit portion of each character in the field.

## Hex Literal Conversion

When a hex literal is associated with a field of type ALPHA or ALPHA NUMERIC, each pair of hex characters is interpreted as one alphanumeric character. When a hex literal is used with an INTEGER field and the value is greater than @7FFFFFFFFF@, the value of the literal is truncated by discarding the high-order characters.

If a hex literal is specified with a BIT field and has a value that contains more significant bits than those specified for the bit field, then the value of the literal is truncated by discarding the high-order characters.

If a hex literal specifies a value that is smaller than the specified field, then hexadecimal zeros are added to the most significant, or left, end of the literal to ensure the proper length.

## Integer Literal Conversion

When an integer literal is associated with a field of type ALPHA NUMERIC or PACKED, the arithmetic value of the integer literal is first converted to a graphic literal or a hex literal. This new literal has the same number of digits as the original integer literal and, if necessary, the character for a SEPARATE sign is added. Any explicitly specified leading zeros in the integer literal are retained when the length of the constructed graphic literal or hex literal is determined.

The length of the constructed literal is used as the default length of the field associated with the literal if no specific field length is specified. However, if an explicit field length is given, the constructed literal can be truncated if it is too long, or padded if it is too short. When truncation takes place, the high-order (leftmost) characters are discarded. When padding takes place, zero characters are added to the high-order (or leftmost) end. If the field is of type SIGNED IS LEADING, the sign is properly preserved after either truncation or padding.

Table 2-2 illustrates which literals are allowed in each field. An X indicates allowable combinations; all other combinations result in syntax errors.

Table 2-2. Allowable Literal Types

Field Type	Literal Type			
	Graphic	Hex	Integer	Real
ALPHA	X	X		
ALPHA NUMERIC	X	X	X	
PACKED		X	X	

continued

Table 2-2. Allowable Literal Types (cont.)

Field Type	Literal Type			
	Graphic	Hex	Integer	Real
ZONE	X	X		
DIGIT	X	X	X	
HEX		X		
INTEGER		X	X	
REAL			X	X
DOUBLE			X	X
BIT		X	X	



# Section 3

## SORT Statements

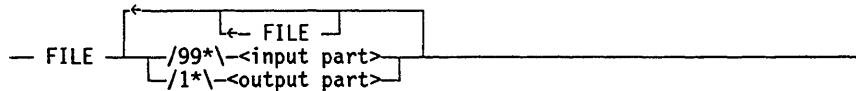
This section describes the required statements for a Sort program. A Sort program must have one <file statement> and one <key statement>, but only one of each.

### FILE Statement

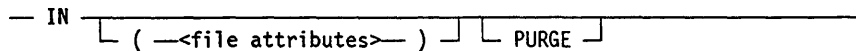
A FILE statement describes and defines both the files to be used as input to a Sort program and the one output file. A Sort program can either sort up to 99 files or merge from two to eight files.

#### Syntax

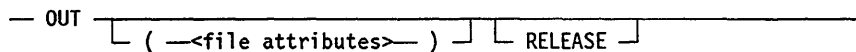
<file statement>



<input part>



<output part>



#### Explanation

<input part>

Describes an input file.

<output part>

Describes the one required output file. By default, the output file assumes all the file attributes of the first input file except the file attributes TITLE, MYUSE, and NEWFILE.

IN

Specifies the definition of an input file. The internal file name of the first input file is IN. Any subsequent internal file names are INn, where n is 2 for the second internal file name and is incremented by 1 for each successive internal file name. For example, a merge of three files involves internal files named IN, IN2, and IN3. However, the file attribute INTNAME can be used to specify a different internal file name.

## SORT Statements

---

< file attributes >

For the syntax and explanation of this option, refer to "File Attributes" in this section.

### PURGE

Closes, purges, and releases the input files to the system. By default, input files are closed with RELEASE. PURGE cannot be specified when the TAGSORT statement is used; otherwise, a syntax error results.

### OUT

Specifies the definition of the output file. The internal file name of the output file is OUT. However, the file attribute INTNAME can be used to specify a different internal file name.

### RELEASE

Returns control of the tape to the system. By default, the file is closed with lock (saved). This option is used only for tape files.

## File Attributes

File attributes appear only in a < file statement >. They can also be specified with the compile and run commands of the Work Flow Language (WFL) and the Command and Edit (CANDE) Message Control System (MCS).

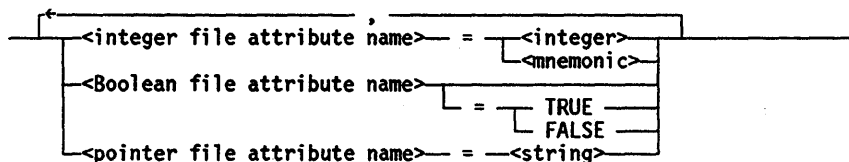
At run time, the record sizes of the input files and that of the output file may not all be the same. The following rules apply:

- The record size of the Sort work file is based on the maximum of all the declared input and output MAXRECSIZE values and the actual MAXRECSIZE of the input file or the maximum of all input merge files.
- If the record size of the output file is not the same as the work file, records are truncated or left-justified with blank fill on the right before being written to the output file.

For more information concerning file attribute values, refer to the "Sort Program Files" appendix. Detailed explanations about each file attribute can be found in the *File Attributes Reference Manual*.

### Syntax

<file attributes>



**Explanation**

< integer file attribute name >

Specifies the attributes that can be expressed as integers or mnemonics.

< Boolean file attribute name >

Specifies the attributes that can be expressed as Booleans. If only the attribute name is listed, its value is set to TRUE.

< pointer file attribute name >

Specifies the attributes that can be expressed as strings. The < string > consists of one or more characters enclosed in quotation marks. The < string > values do not need to end with a period (.).

Sort can read input files with FILEORGANIZATION = NOTRESTRICTED or FILEORGANIZATION = INDEXEDNOTRESTRICTED. Sort output files must have FILEORGANIZATION = NOTRESTRICTED.

Files that have FILEORGANIZATION = RELATIVE or FILEORGANIZATION = INDEXED cannot be used as Sort input or output files. Files with variable-length records cannot be used as Sort input files; furthermore, a Sort program does not produce output files with variable-length records.

**Example: Using the Pointer File Attribute Name**

In the following example, the file attributes TITLE and FAMILYNAME both have < pointer file attribute name > s:

```
MAXRECSIZE = 15,
NEWFILE = TRUE,
KIND = TAPE,
TITLE = "A/B",
FAMILYNAME = "TGFSOFTDOC"
```

**KEY Statement**

A KEY statement contains information defining the sort or merge keys. This information consists of a displacement, a length, a processing order, and a type. Only one KEY statement can appear in a Sort program.

A single KEY statement can contain up to 200 key fields. When multiple key fields are specified, the first one listed is the primary (or most significant) key. Before the sort or merge is performed, all subsequent keys are concatenated to the primary key in the order in which they are listed.



## SORT Statements

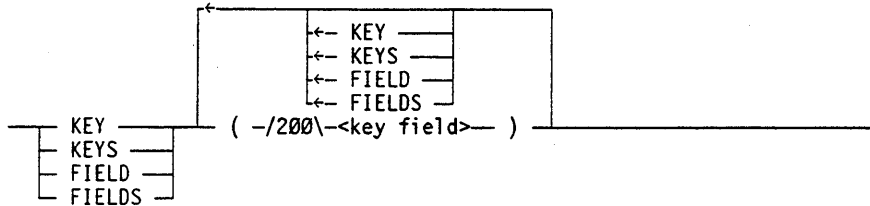
---

The processing of the input files during a sort or merge occurs as follows:

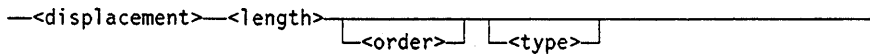
- Records are sorted or merged according to the specifications of the primary key.
- Subsequent keys are used to further sort or merge any records that were identical from the previous key.

### Syntax

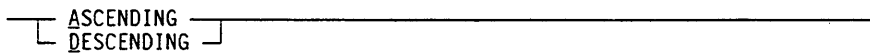
<key statement>



<key field>



<order>



### Explanation

KEY, KEYS, FIELD, FIELDS

Any of these words can be used to signify the start of a KEY statement.

<key field>

Specifies a field within a record to be used to sort the input files.

< displacement >

Specifies the position of the first element in the key field.

< length >

Specifies the number of units in the key field.

< order >

Specifies whether the key is to be sorted in ascending or descending order. ASCENDING is the default order.

ASCENDING order sorts from the lowest value to the highest value. DESCENDING order sorts from highest value to lowest value.

< type >

Refer to the explanation of type in the "Sort Program Structure" section. ALPHA is the default type.

#### **Example: Using a Single Key Field**

The following Sort program uses a single key to sort the input file. The field type is ALPHA (by default). The key begins at position 3 of each record and has a length of 1. The sort is performed in ascending order.

```
DISKSORT
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (3 1 A)
```

#### **Example: Using Multiple Key Fields**

In the following Sort program, input records are to be sorted using multiple key fields, each of type ALPHA. The primary key begins at position 4, has a length of 2, and uses descending order. Any records having identical primary key fields are then sorted based on the second key, which begins in position 10, has a length of 3, and uses ascending order. The third key is then used to sort any remaining unresolved identical records in ascending order based on position 20.

```
DISKSORT
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (4 2 D) (10 3 A) (20 1 A)
```

The previous Sort program can also be written in the following form:

```
DISKSORT
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (4 2 D)
    (10 3 A)
    (20 1 A)
```



# Section 4

## Optional SORT Statements

This section describes the optional SORT statements that can appear in Sort programs. These statements are divided into three types:

- SORT processing statements
- SORT parameter statements
- Record selection statements

### SORT Processing Statements

SORT processing statements specify whether a merge or a specific type of sort is to be performed. SORT processing statements are optional; however, some defaults are assumed by the compiler if a Sort program lacks a processing statement. When a Sort program has only one input file, a disk sort is assumed; when the program specifies two or more input files, a merge is assumed.

### COLLATE Statement

The COLLATE statement specifies that a Sort program is to use the system default internationalization collating sequence when the program is run. This optional statement initiates a collating sequence in which records are sorted or merged by any type of sort, enabling the compiler to handle correctly the sort and compare operations for the alphanumeric literals.

#### Syntax

<collate statement>

— COLLATE —————

#### Example

The following example shows a program that uses the system default internationalization collating sequence to sort an input file on disk.

```
COLLATE
DISKSORT
FILE IN(TITLE="IN",KIND=DISK,DEPENDENTSPECS)
FILE OUT(TITLE="OUTPUT",KIND=DISK,DEPENDENTSPECS)
KEY (1 10)
```

### DISKANDTAPE Statement

An Integrated Tape and Disk (ITD) sort is performed when the DISKANDTAPE statement is used. An ITD allows input files to come from both tape and disk sources.

The disk pack family used for the temporary disk sort space is normally the family DISK. This allocation can be modified by using the WORKFAMILY statement. (For more information, refer to the WORKFAMILY and WORKSIZE statements in this section.)

Only one DISKANDTAPE statement can be specified in a Sort program. A syntax error results if the DISKANDTAPE statement is specified in conjunction with any of the following SORT statements:

- MERGE
- DISKSORT
- TAPESORT
- MEMORYONLY

#### Syntax

<diskandtape statement>

DISKANDTAPE <integer>  
ITD

#### Explanation

<integer>

Specifies the number of tapes to be used for the ITD sort. This number must be between 3 and 8, inclusive.

#### Example

In the following example, three Sort work tapes are used. The input file is sorted using an INTEGER key field in position 2 for a length of 1, in ASCENDING order. (The length must be 1 for type INTEGER.)

```
DISKANDTAPE 3
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (2 1 INTEGER)
```

### DISKSORT Statement

The DISKSORT statement specifies that a disk sort is to be performed, meaning that the temporary file space required for the SORT procedure is allocated on disk. The disk pack family used for this temporary file space is DISK; however, this specification can be modified by using the WORKFAMILY statement. (Refer to the WORKFAMILY statement in this section.)

Only one DISKSORT statement can be specified in a Sort program. A syntax error results if the DISKSORT statement is specified in conjunction with any of the following SORT statements:

- DISKANDTAPE
- MEMORYONLY
- MERGE
- TAPESORT

### Syntax

<disksort statement>

— DISKSORT \_\_\_\_\_|

### Example

The following example shows a program that sorts the input file on disk. Note that because there is only one input file, a disk sort would be assumed by the SORT compiler if the DISKSORT statement had been omitted. Also, because ALPHA and ASCENDING are default values, the key statement could be written as KEY (2 3).

```
DISKSORT
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (2 3 A ALPHA)
```

## Optional SORT Statements

---

## MEMORYONLY Statement

The MEMORYONLY statement specifies that a sort is to be performed completely in memory, without using disk or tape work files.

Only one MEMORYONLY sort statement can be specified in a Sort program. A syntax error results if the MEMORYONLY statement is specified in conjunction with any of the following SORT statements:

- DISKANDTAPE
- DISKSORT
- TAPESORT
- MERGE

The WORKSIZE statement has no effect when specified with the MEMORYONLY sort statement; if it appears, the compiler ignores it.

### Syntax

<memoryonly statement>

— MEMORYONLY \_\_\_\_\_|

### Example

In the following example, an input file residing on tape is to be sorted in main memory, using a PACKED key field in position 10 for a length of 4, in ASCENDING order:

```
MEMORYONLY
FILE IN (TITLE = "INPUT", KIND = TAPE)
FILE OUT (TITLE = "OUTPUT")
KEY (10 4 A PACKED)
```

## MERGE Statement

The MERGE statement specifies that a merge is to be performed on the input files. At least two but not more than eight input files must be declared; otherwise, a syntax error results. A merge is assumed by the SORT compiler if more than one input file is specified in the Sort program and it contains no other SORT processing statements.

Merges cannot be made more efficient by including any of the SORT parameter statements in the Sort program.



## Optional SORT Statements

---

Only one MERGE statement can be specified in a Sort program. A syntax error results if the MERGE statement is specified in conjunction with any of the following SORT statements:

- DISKANDTAPE
- MEMORYONLY
- DISKSORT
- TAPESORT
- TAGSORT
- TAGSEARCH
- STABLE

### Syntax

<merge statement>

— MERGE —

### Examples

This example shows a Sort program that specifies a merge is to be performed on the four input files using an ALPHA key field starting at position 2. Because there is more than one input file, the MERGE statement does not have to be included.

```
MERGE
FILE IN (TITLE = "INPUT1")
      IN (TITLE = "INPUT2")
      IN (TITLE = "INPUT3")
      IN (TITLE = "INPUT4")
FILE OUT (TITLE = "OUTPUT")
KEY (2 3 A ALPHA)
```

In the following example, the default values for the input files and the key statement are assumed. In regard to the input files, the SORT compiler will search for files named IN, IN2, IN3, and IN4. Usually this type of program construct is used when the same program will be used with many different files, the names of which can be specified with the RUN statement through file attribute equations.

```
FILE IN
      IN
      IN
      IN
FILE OUT (TITLE = "OUTPUT")
KEY (2 3)
```

## STABLE Statement

The STABLE statement causes records that have equal key values to be placed into the sorted output file in the order in which they appeared in the input file. Equal records are

defined as having identical key fields. If a STABLE statement is not specified, the order of equal records in the output file cannot be predicted.

During the input phase, an extra implicit key is appended to the keys to be used to determine the sorting order for records having equal keys. The extra implicit key has the value of the relative record number in the input file. During the output phase, this implicit key is removed.

Only one STABLE statement can be specified in a Sort program, and a STABLE statement cannot be used with a MERGE statement; otherwise, a syntax error results.

### Syntax

<stable statement>

— STABLE \_\_\_\_\_|

## TAGSORT Statement

The TAGSORT statement sorts a file and creates an output file containing indexes that point to the relative locations of records in the original file. An index value of 0 indicates the first record in the original file.

Only one TAGSORT statement can be specified in a Sort program. A syntax error results if the TAGSORT statement is specified in conjunction with any of the following SORT statements:

- MERGE statement
- TAGSEARCH statement

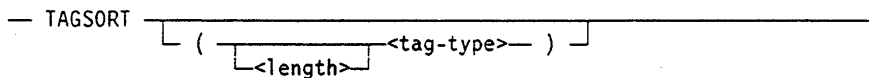
If a TAGSORT statement is specified, there must be only one input file. The input file need not be a disk file, but the usefulness of indexes to a nondisk file is limited.

When the TAGSORT statement is specified, the MAXRECSIZE of the output file is changed to <length>. The value of the UNITS attribute is changed to reflect the units of <tag-type>. The FILEKIND of the output file is set to DATA. The BLOCKSIZE of the output file is changed to the largest value that is a multiple of <length> and less than 2520 bytes.

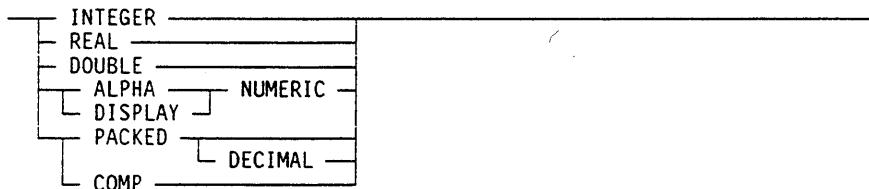
If an overflow occurs when the index value is created, the sort aborts with a run-time error.

**Syntax**

<tagsort statement>



<tag-type>



**Explanation**

< length >

Specifies the size of the indexes to be created. The default index size is 1. If both length and tag-type are omitted, the resulting output file is in the proper form for a Report Program Generator (RPG) ADDRROUT file.

< tag-type >

Specifies the index < type >. Refer to the explanation of type in the "Sort Program Structure" section for a complete explanation. (The explanation for < type > is the same as for < tag-type >.) The default value for tag-type is INTEGER.

**Example**

The following example shows a Sort program that creates an output file containing indexes (of tag-type INTEGER and length 3) of the relative record locations within the input file:

```
DISKSORT
TAGSORT (3 INTEGER)
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (3 1 D COMP)
```

**TAPESORT Statement**

The TAPESORT statement specifies that a tape sort is to be performed, meaning that the temporary file space required for the Sort procedure is allocated on tape.

Only one TAPESORT statement can be specified in a Sort program. A syntax error results if the TAPESORT statement is specified in conjunction with any of the following SORT statements:

- DISKANDTAPE
- MEMORYONLY

- DISKSORT
- MERGE

The WORKSIZE statement is ignored by the SORT compiler if it is specified in a Sort program containing a TAPESORT statement.

For example, if tape is specified as both the input and the output medium, and the system has exactly the same number of tape drives as are specified for use as work files, one of those tape drives is available for use in the final output pass. If there are three tape drives on the system, the sort input can come from tape and the sort output can go to tape while still using only three Sort work tapes.

### Syntax

<tapesort statement>

— TAPESORT —<work tapes>—————|

<work tapes>

— <integer> —————|

### Explanation

< work tapes >

Specifies an integer value indicating the number of work tapes to be used during the TAPESORT procedure. The number of work tapes must be between 3 and 8, inclusive; otherwise, a syntax error results.

### Example

The following example program uses the TAPESORT procedure, with three work tapes, to sort the input file, which resides on tape:

```
TAPESORT 3
FILE IN (TITLE = "INPUT", KIND=TAPE)
FILE OUT (TITLE = "OUTPUT")
KEY (4 1)
```

## SORT Parameter Statements

SORT parameter statements are used to increase sorting efficiency.

### MEMORY Statement

The MEMORY statement allocates additional memory to the compiled Sort program.

Each type of Sort procedure has a memory size default value based on an expected maximum record size. For a disk sort, memory for 1200 records is requested. For a

## Optional SORT Statements

---

tape sort, memory for 200 records for each work tape is requested. For a tape and disk sort, memory for 600 records plus 200 records for each work tape is requested. For a memory-only sort, memory for the entire sort input is requested.

Only one MEMORY statement can be specified in a Sort program; otherwise, a syntax error results.

Increasing the memory available to the Sort procedure is the most significant way to increase sorting efficiency. However, if the memory size is increased beyond the optimum size, the sort might run slower rather than faster. The optimum memory size to allocate varies according to file size, which involves the record size, the number of records, and the blocking factor. The number of other jobs in the mix at the time of the sort can also affect the optimum memory size. Information about how to calculate optimum memory sizes for the various Sort procedures can be found in the *System Software Utilities Operations Reference Manual*.

### Syntax

<memory statement>

— MEMORY —<integer> — WORDS —

### Explanation

<integer >

Specifies the number of words to which memory is increased. The integer value must be within the range 3,500 through 1,000,000.

WORDS

Used for program documentation. WORDS is ignored by the SORT compiler.

## PARITY DISCARD Statement

The PARITY DISCARD statement causes input file records to be discarded if an irrecoverable parity error is detected while the input file is being read. If the PARITY DISCARD statement is specified, all the records in the block of data in which the error occurred are discarded. The Sort program displays notification of each record discarded and its relative record position in the input file. The program is discontinued if a parity error occurs when the PARITY DISCARD statement has not been specified.

When the TAGSEARCH statement is used with the PARITY DISCARD statement in a Sort program, input records having parity errors are discarded during the first pass in which the file is read. During the output phase of the sort, when the input records are read again, any parity error detected will abort the Sort program.

Only one PARITY DISCARD statement can be specified in a Sort program; otherwise, a syntax error results.

**Syntax**

&lt;parity discard statement&gt;

— PARITY — DISCARD —————|

**RECORDS Statement**

The RECORDS statement optimizes the sort operation by supplying an estimate of the total number of records to be sorted.

Only one RECORDS statement can be specified in a Sort program; otherwise, a syntax error results. The RECORDS statement is ignored by the compiler if it appears in conjunction with the WORKSIZE statement or the MERGE statement.

**Syntax**

&lt;records statement&gt;

— RECORDS —&lt;record estimate&gt;—————|

**Explanation**

&lt;record estimate&gt;

Estimates the number of records to be sorted after the application of an INCLUDE and DELETE statement. If a record estimate is not specified and the input file is on disk, the value is supplied from the LASTRECORD file attribute. If a record estimate is not specified and the file does not reside on disk, a default value of 20,000 records is used.

The record estimate must not be smaller than the actual number of records or the Sort program will fail at run time. Alternately, making the record estimate too large increases the size of the work file and causes the sort to use more disk space than necessary.

**TAGSEARCH Statement**

The TAGSEARCH statement performs a tag sort. The resulting indexes are used to build a sorted output file. A TAGSEARCH sort can attain a greater sorting speed than other types of sorts depending on the number of records to be sorted and other characteristics of the file.

A TAGSEARCH statement can accept only one input file, and that file must be a disk file. Only one TAGSEARCH statement can be specified in a Sort program, and it cannot be specified in conjunction with a MERGE statement or a TAGSORT statement; otherwise, a syntax error results.

**Syntax**

&lt;tagsearch statement&gt;

— TAGSEARCH —————|

### Example

The following example shows the inclusion of the TAGSEARCH statement in a Sort program. A tagsearch can be specified with any of the Sort processing options except the MERGE statement. There must be only one input file.

```
DISKSORT
TAGSEARCH
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (8 1 D REAL)
```

### WORKFAMILY Statement

The WORKFAMILY statement assigns a specific disk pack family for use by the Sort program work files.

Only one WORKFAMILY statement can be specified in a Sort program; otherwise, a syntax error results. The WORKFAMILY statement is ignored by the compiler if used with any of the following:

- MEMORYONLY sort statement
- TAPESORT statement
- MERGE statement

### Syntax

<workfamily statement>

— WORKFAMILY —<family name>—————|

### Explanation

< family name >

Specifies the name of a disk pack family to be used by the Sort program work files.

### WORKSIZE Statement

The WORKSIZE statement specifies the number of words to use for the Sort work file. By default, the SORT compiler calculates the size of the work file based on the number of input file records. However, if a DISKANDTAPE sort is specified without an accompanying WORKSIZE statement, the work file size defaults to 900,000 records.

Only one WORKSIZE statement can be specified in a Sort program; otherwise, a syntax error results. The RECORDS statement is ignored by the compiler when declared with the WORKSIZE statement.

**Syntax**

<worksize statement>

— WORKSIZE —<integer> —————  
                  └── WORDS ─┘

**Explanation**

<integer >

Refer to the explanation of integer in the “Sort Program Structure” section.

**WORDS**

Used for program documentation. WORDS is ignored by the SORT compiler.

## INCLUDE and DELETE Record Selection Statements

The record selection statement (INCLUDE and DELETE statement) defines which records are to be included or excluded from the input files being sorted or merged.

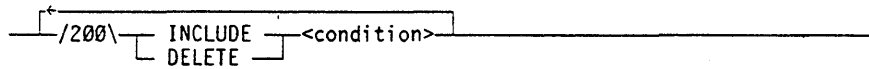
The INCLUDE and DELETE statement allows the inclusion and exclusion of certain records by comparing portions of each record either with other portions of the same record or with specified literals. If multiple INCLUDE and DELETE actions are specified (up to 200 are allowed), each successive INCLUDE or DELETE action creates further subsets of the records selected by previous INCLUDE or DELETE actions.



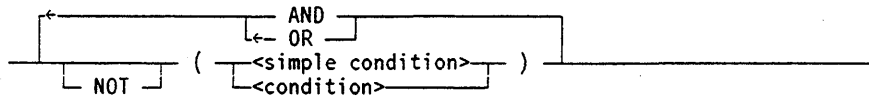
## Optional SORT Statements

### Syntax

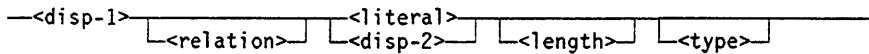
<include and delete statement>



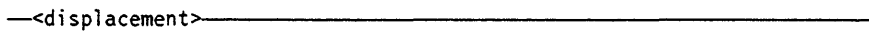
<condition>



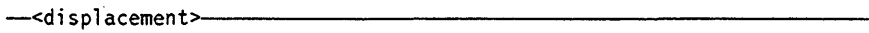
<simple condition>



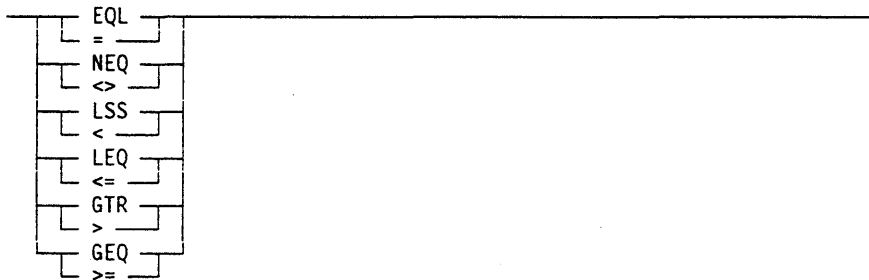
<disp-1>



<disp-2>



<relation>



### Explanation

<condition>

Specifies the conditions under which records are included or excluded. The precedence of operators is NOT, AND, OR, and <relation>, in that order. The evaluation of operators proceeds from left to right. A <simple condition> within parentheses is evaluated before the remainder of the <condition> is evaluated.

<simple condition>

Specifies the operands and the relations used to compare portions of the record with other portions of the same record or with specified literals.

<relation>

Specifies a relational operator for comparing operands. Valid entries are EQL (equal), NEQ (not equal), GTR (greater than), GEQ (greater than or equal), LSS (less than), and LEQ (less than or equal). The default value is EQL.

< literal >

Refer to the explanation of literal in the "Sort Program Structure" section.

< length >

Specifies the length of the operands to be compared. If < disp-2 > is used, the default length is 1. If < literal > is used, the default length is the length of the literal. If an explicit type is not given, the length represents bytes. (Refer to the explanation of type in the "Sort Program Structure" section.)

< type >

Refer to the explanation of type in the "Sort Program Structure" section.

< disp-1 >, < disp-2 >

Expresses the first or second operand in a simple condition as a displacement.

### Examples

The following example shows how displacements are used to compare portions of each input file record with other portions of the same input file record. All records in the input file in which the real value in word 8 does not equal the real value in word 10 or word 11 are deleted. Then, all records in which the real value in word 2 is greater than the real value in word 5 are included. The records fitting these restrictions are sorted using a REAL key field in word 2, for a length of 1, in ascending order.

```
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (WORD 2 1 REAL)
DELETE ((WORD 8 = WORD 10 REAL) OR (WORD 8 = WORD 11 REAL))
INCLUDE (WORD 2 GTR WORD 5 REAL)
```

The next example shows how literals are used to compare portions of each input file record with other portions of the same input file record. All input file records having an alphanumeric value in character position 5 greater than 3 and less than or equal to 9 are included. Then, all records in which the alphanumeric value in positions 10 through 12 is equal to AAA are deleted. The records fitting these restrictions are sorted using an ALPHA key field in character position 2 for a length of 3, in descending order.

```
DISKSORT
FILE IN (TITLE = "INPUT")
FILE OUT (TITLE = "OUTPUT")
KEY (2 3 D ALPHA)
INCLUDE ((5 GTR "3") AND (5 LEQ "9"))
DELETE (10 EQL "AAA" 3)
```



# Section 5

## **SORT Compiler Control Records**

Compiler control records (CCRs) provide a mechanism by which the programmer controls options provided by the compiler. These options fall into one of six categories, and control the following:

- Source language inputs
- Source language output
- Optional compilation mechanisms
- Printed outputs
- Compiler diagnostic messages
- Compiler debugging

A CCR contains compiler control statements made up of options or groups of options and their associated parameters (if any). If no compiler control statements appear on a CCR, the CCR is considered null.

### **Types of Compiler Control Records**

CCRs are of two types:

- Permanent CCRs, which may remain associated with the source language
- Temporary CCRs, which are only relevant to a given compilation

### **Types of Options**

Compiler control options are of three types:

- Boolean
- Immediate
- Value

A Boolean option is either SET or RESET. (SET is synonymous with TRUE, RESET with FALSE.) When the value of a Boolean option is TRUE, the SORT compiler applies the associated function to all subsequent processing until the value of the Boolean option is changed to FALSE. Boolean options can also have associated parameters, which are related to the function the Boolean option affects.

An immediate option causes the compiler to apply a function that is independent of subsequent processing. Immediate options can also have associated parameters.

A value option causes the compiler to store a value associated with a given function.

## Using TASKSTRING to Submit Compiler Control Records

The SORT compiler also accepts compiler control records supplied by the TASKSTRING task attribute at compile time. The following WFL example sets the ERRORLIST and ERRORLIMIT options for a SORT compilation:

```
COMPILE OBJECT/SORTPROG WITH SORT LIBRARY GO;  
COMPILER FILE CARD (TITLE = ALPHASORT ON DPMST);  
COMPILER TASKSTRING = "$SET ERRORLIST ERRORLIMIT = 20";
```

## Compiler Control Record Syntax

CCRs are interpreted from left to right, beginning at the first text position following the dollar sign (\$) and continuing until the last text position.

The keywords SET, RESET, and POP affect the setting of Boolean options. Each of the Boolean options has an associated stack in which up to 48 previous values of the option are saved. The management of the current value and the stack of previous values of the option are described in the following paragraphs.

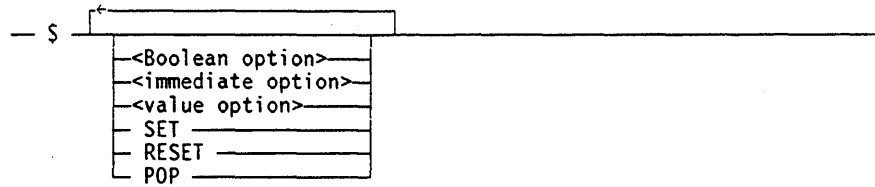
Depending on the syntax used, the status of Boolean options is determined in one of the following three ways:

- If a Boolean option appears on a CCR and is not the object of an explicit SET or RESET, then it is implicitly set to TRUE. The previous setting of the option is pushed onto the top of the stack.
- If a Boolean option appears on a CCR as the object of an explicit SET or RESET, then the specified Boolean option is set to TRUE or FALSE, respectively. The previous setting of the option is pushed onto the top of the stack.
- If the immediate option CLEAR appears on a CCR, then all Boolean options are set to FALSE, except for the MERGE option, SINGLE option, and, conditionally, the NEW option. All previously stacked settings are discarded. Refer to "CLEAR (Immediate)" in this section.

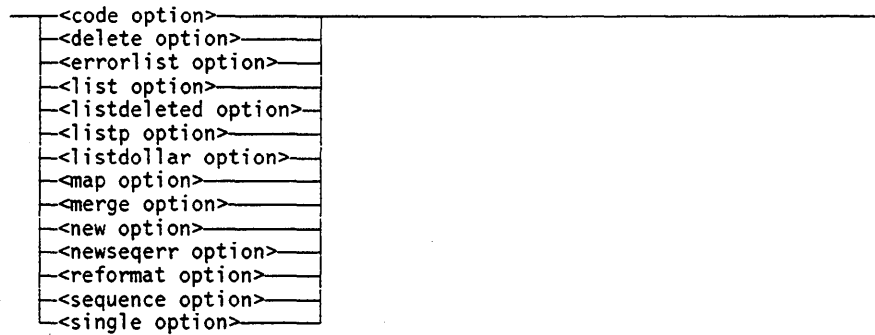
Individual compiler control options are further defined under "Compiler Control Options" in this section.

**Syntax**

<compiler control record>



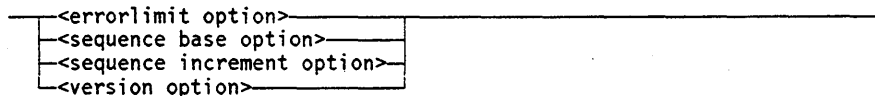
<Boolean option>



<immediate option>



<value option>



**Explanation**

\$

A dollar sign (\$) in column 1 indicates a temporary CCR; in column 2 it indicates a permanent CCR. Placing a \$ in both columns 1 and 2 is not allowed.

< Boolean option >

An option that has a value of TRUE or FALSE.

< immediate option >

An option causing the compiler to apply a function that is independent of subsequent processing.

< value option >

An option that causes the compiler to store a value associated with a given function.

### SET/RESET

Assigns a value of TRUE or FALSE, respectively, to a given Boolean option. The current value of the option is pushed onto the stack for that option.

### POP

Assigns the next stack value to a Boolean option and discards the current setting. Each Boolean option has an associated stack. When a Boolean option is set to TRUE or FALSE, either explicitly by a SET or RESET or implicitly, then the previous setting is pushed onto the stack.

## Compiler Control Options

The options that can appear on CCRs are defined as follows.

### CLEAR (Immediate)

The CLEAR option causes the compiler to disable (set to FALSE) all Boolean options except the MERGE option, the SINGLE option, and, conditionally, the NEW option.

If a source-language image has been written to the new symbolic file (NEWSOURCE) because the NEW option is TRUE, then the NEW option is not set to FALSE.

#### Syntax

<clear option>

— CLEAR —————|

### CODE (Boolean)

The CODE option causes the compiler to produce a listing of the object code produced by the compilation. The code listing is generated after the successful compilation of the SORT statements. A code listing is not produced if syntax errors result. Also, only the most recent setting of the CODE option has an effect because the code listing is produced at the end of the compilation. The LIST option must be TRUE to produce the listing. The default value for the CODE option is FALSE.

#### Syntax

<code option>

— CODE —————|

### DELETE (Boolean)

The DELETE option causes the compiler to discard source-language images from the secondary input file (SOURCE) until the option is set to FALSE.

This option can only appear on a CCR in the primary source-language input file (CARD).

This option is ignored if the MERGE option is FALSE. The DELETE option does not alter the normal merging process; however, it causes the compiler to unconditionally discard all source-language images selected from the secondary input file, including CCRs. This discarding of SOURCE file records begins with the first record in the SOURCE file that has either the same sequence number as the CARD file record that sets the DELETE option to TRUE or the next higher sequence number. The last record discarded from the SOURCE file has either the same sequence number as the CARD file record that sets the DELETE option to FALSE or the next lower sequence number.

The source-language records discarded when this option is set to TRUE are not carried forward to the output symbolic file (NEWSOURCE) if the NEW option is TRUE. In addition, these records are not listed unless the LISTDELETED option is TRUE.

The default value for the DELETE option is FALSE.

**Syntax**

```
<delete option>
— DELETE _____|
```

**ERRORLIMIT (Value)**

The ERRORLIMIT option specifies the maximum number of errors for the compiler to detect before the compilation is terminated.

The default value for the ERRORLIMIT option is 10 if the program is initiated through Command and Edit (CANDE) and 100 if the program is initiated through the Work Flow Language (WFL). A value of 0 causes the compiler to tolerate an unlimited number of errors.

If the error limit is exceeded, the compiler produces a listing of the errors and informs the user that the compilation was terminated because the error limit was exceeded.

If the error limit is exceeded and the NEW option is TRUE, the new file being created is locked. (Refer to the explanation of “NEW (Boolean)” in this section.)

**Syntax**

```
<errorlimit option>
— ERRORLIMIT [ = ] <integer> _____|
```

**Explanation**

<integer >

Any unsigned integer.



=

Used only for program documentation. The equal sign (=) is ignored by the SORT compiler.

### ERRORLIST (Boolean)

The ERRORLIST option controls the destination of the error messages that are generated by the SORT compiler.

If the ERRORLIST option is TRUE, the compiler error messages go to a file entitled either ERRORFILE or ERRORS. If the SORT compiler was invoked from CANDE, the destination file for the compiler error messages is entitled ERRORS. The error messages also go to the LINE file if a LINE file is used. The LINE file is in use if either the LIST option, LISTDELETED option, LISTP option, or LISTDOLLAR option is TRUE when the error is detected, or if earlier in the compilation, any of these options were TRUE and then set to FALSE. If the LINE file is not in use when the compiler detects the error condition, the error message does not go to the LINE file.

The default setting for the ERRORLIST option is FALSE for Sort compilations initiated from WFL, and TRUE for compilations initiated from CANDE. The default setting for the ERRORFILE (or ERRORS) file for the file attribute KIND is PRINTER for compilations initiated from WFL and REMOTE for compilations initiated from CANDE.

#### Syntax

<errorlist option>

— ERRORLIST \_\_\_\_\_

### GSORT (Boolean)

The GSORT option makes sort programs compatible with the industry-standard GSORT syntax and capabilities. (The program must be coded in a fixed, column-oriented format similar to programs coded in RPG.)

The GSORT option defaults to FALSE for SORT compilations submitted from CANDE or WFL.

Once the GSORT option is set, it cannot be reset. Any subsequent statements that reset GSORT are treated as errors.

When the GSORT option is set to TRUE, the following additional sort features are available:

- Output records can be remapped to have a different format than the input records. For example, the remapped output records can be constructed out of selected parts of an input record.
- Multiple sets of input records can be selected according to specific criteria. Different sorting and remapping criteria can be applied to each set.

- Special characters can be used to force related types of records to be grouped together in the output.
- Alternate collating sequences can be specified to permit sorting of records in an order different from that of the standard EBCDIC collating sequence. You can change the collating sequence for all or selected characters.
- Sorted data can be totaled by adding up fields in records.
- Data can be selected and an output file can be created without performing a sort.
- Records selected from a single file can be sorted to produce a record number file that reflects the new order, similar to a tag sort.

See the appendix “Using GSORT to Code and Run Sort Programs” for details of how to design, code, and run Sort programs using the GSORT option.

**Syntax**

<gsort option>

— GSORT —————|

**LIST (Boolean)**

If the LIST option is TRUE, the SORT compiler produces a listing of the CARD file input records in the LINE file. If the MERGE option is TRUE, the SORT compiler also produces a listing of the SOURCE file records. Any CCRs having a dollar sign (\$) in column 1 are not listed. (Refer to “LISTDOLLAR (Boolean)” in this section.) However, if the LIST option is set to TRUE, the CCRs having a dollar sign in column 2 are listed. Records from the SOURCE file that have been discarded, either by the action of the DELETE option or by a matching record from the CARD file, are not listed unless the LISTDELETED option is also TRUE.

If both the LIST option and the ERRORLIST option are FALSE, all the error messages in the LINE file are listed. If the LIST option is FALSE and the ERRORLIST option is TRUE, the error messages are written to the ERRORFILE (or ERRORS) file. If both the LIST option and the ERRORLIST option are TRUE, the error messages are written to both the LINE and the ERRORFILE (or ERRORS) files.

The default value for the LIST option is FALSE for compilations initiated from CANDE, and TRUE for compilations initiated from WFL.

**Syntax**

<list option>

— LIST —————|

**LISTDELETED (Boolean)**

The LISTDELETED option causes the compiler to produce a listing of the source-language input records that are deleted when the DELETE option is TRUE. The deleted source-language input records are identified on the listing with a D.

The default value for the LISTDELETED option is FALSE.

### Syntax

<listdeleted option>

— LISTDELETED —————|

## LISTDOLLAR (Boolean)

If the LISTDOLLAR option and the LIST option are TRUE, the compiler produces a listing of all temporary CCRs encountered during the compilation. The temporary CCRs are identified by a dollar sign (\$) in column 1.

The default value for the LISTDOLLAR option is FALSE.

LIST\$ is a synonym for LISTDOLLAR.

### Syntax

<listdollar option>

— LISTDOLLAR —————|  
— LIST\$ —————|

## LISTP (Boolean)

The LISTP option causes the compiler to produce a listing of the source-language records that appear in the primary input file (CARD). Source-language records from the secondary input file (SOURCE) are not listed.

If the LIST option is TRUE, then the LISTP option has no effect. The default value for the LISTP option is FALSE.

### Syntax

<listp option>

— LISTP —————|

## MAP (Boolean)

The MAP option causes the compiler to include, as part of the output listing, address couples that indicate the storage locations of variables within the code file produced by the compilation. The MAP option has no effect if the LIST option is FALSE.

The default value for the MAP option is FALSE.

**Syntax**

<map option>

— MAP —————

**MERGE (Boolean)**

The MERGE option controls how the SORT compiler handles the secondary source-language input file, SOURCE. The compiler reads only from the input file, CARD, if the MERGE option is FALSE.

If the MERGE option is TRUE, records from the CARD file are merged with the SOURCE file input records. The merge is based on the value of the sequence number field (columns 73 through 80) in the input records. A record is taken from the SOURCE file if the sequence number is less than the sequence number of the next record in the CARD file.

However, if both files contain records that have identical sequence numbers, the record from the CARD file is used. The record from the SOURCE file is then discarded. Also, whole groups of records can be discarded from the SOURCE file by using the DELETE option.

All records in the SOURCE and CARD files must have ascending sequence numbers for the merge action to work correctly.

On the CCR, the keyword MERGE can be followed by a file title. This title is the disk file title of the SOURCE file to be used. Alternatively, the disk file title of the SOURCE file can be specified to the compiler by using either a WFL or a CANDE file attribute equation.

The default value for the MERGE option is FALSE.

**Syntax**

<merge option>

— MERGE —————  
└<file title>

**Explanation**

<file title>

A quoted string specifying the name of the file containing secondary source-language records. If <file title> is not specified, then SOURCE is assumed unless pre-empted by a file equation.

**NEW (Boolean)**

The NEW option generates a disk file containing the input records read by the SORT compiler. If the value of this option is FALSE, a NEWSOURCE file is not generated.

If the value of the NEW option is TRUE, the input records of the compiler are written to the NEWSOURCE file. Temporary CCRs and the records discarded from the SOURCE file are not written to the NEWSOURCE file. This NEWSOURCE file can then be used as the SOURCE file for compiling the Sort program in the future. The NEWSOURCE file is created and locked on disk even if syntax errors are detected by the SORT compiler, or if the compiler aborted the compilation because the specified ERRORLIMIT value was exceeded.

If the REFORMAT option is TRUE, the records that are written to the NEWSOURCE file are modified to use A Series SORT compiler syntax rather than B 1000 syntax. New sequence numbers can be given to records in the NEWSOURCE file by using the SEQUENCE option, <sequence base option>, and <sequence increment option>.

On a CCR, the keyword NEW can be followed by a file title. This title is the disk file title of the created SOURCE file. Alternatively, the disk file title of the NEWSOURCE file can be specified to the compiler by using either a WFL or a CANDE file attribute equation.

The default value for the NEW option is FALSE.

### Syntax

<new option>

— NEW —————  
      └─<file title>┘

### Explanation

<file title>

A quoted string specifying the name of the symbolic file to which accepted records are written. If <file title> is not specified, then NEWSOURCE is assumed unless pre-empted by a CANDE or WFL file attribute equation.

## NEWSEQERR (Boolean)

If TRUE, the NEWSEQERR option interacts with the NEW option to ensure that the records written to the NEWSOURCE file have sequence numbers (in columns 73 through 80) that are strictly in ascending order. A syntax error is given for each occurrence of a record in the NEWSOURCE file that is not in ascending order. The NEWSOURCE file is locked on disk even if sequence errors are detected in the file.

The default value for the NEWSEQERR option is FALSE.

### Syntax

<newseqerr option>

— NEWSEQERR —————

## REFORMAT (Boolean)

The REFORMAT option causes acceptable (to the SORT compiler) B 1000 Sort syntax to be translated into A Series SORT compiler syntax in the NEWTAPE and LINE files. Even A Series SORT compiler syntax can be reformatted; the result is that statements are reordered, and missing defaults are replaced with explicit syntax items.

If used, the REFORMAT option must be TRUE before the beginning of the source program and remain TRUE throughout the compilation.

If the LIST option is TRUE, the reformatted B 1000 syntax is written to the LINE file following the listing of the unmodified input records. If the NEW option is TRUE, the reformatted text (instead of the unmodified input records) is written to the NEWSOURCE file.

The default value for the REFORMAT option is FALSE.

### Syntax

<reformat option>

— REFORMAT \_\_\_\_\_|

## SEQUENCE (Boolean)

The SEQUENCE option causes the compiler to assign new sequence numbers to the source-language images accepted for compilation.

This option assigns new sequence numbers to the NEWSOURCE file. The sequence numbers are based on the values of the <sequence base option> and <sequence increment option>.

New sequence numbers are assigned to input records as follows:

- After the record has been accepted by the merge action if the MERGE option is TRUE, and before the record is written to the LINE file if the LIST option is TRUE
- Before the sequence number order is checked if the NEWSEQERR option is TRUE
- Before the record is written to the NEWSOURCE file if the NEW option is TRUE

The SEQUENCE option affects only input source-language records that the compiler encounters after the merging process.

The default value for the SEQUENCE option is FALSE.

### Syntax

<sequence option>

— SEQUENCE \_\_\_\_\_|

## Sequence Base (Value)

The < sequence base option > contains the sequence number that is assigned to the next source-language record when the SEQUENCE option is TRUE. After each record is sequenced, the value of the < sequence base option > is increased by the value of the < sequence increment option > .

The default value of the < sequence base option > is 1000.

### Syntax

<sequence base option>

— /5\-<digit> —————

## Sequence Increment (Value)

The value of the < sequence increment option > increments the value of the < sequence base option > when records are being sequenced because the SEQUENCE option is TRUE.

The default value of the < sequence increment option > is 1000.

### Syntax

<sequence increment option>

— + /5\-<digit> —————

## SINGLE (Boolean)

If TRUE, the SINGLE option single-spaces the listing in the LINE file. If the SINGLE option is FALSE, the listing is double-spaced.

The default value for the SINGLE option is TRUE.

### Syntax

<single option>

— SINGLE —————

## VERSION (Value)

The VERSION option specifies a Sort program version number to be used during compilation for setting the version information field (columns 83 through 90). If the version information field contains a < patch number > and a VERSION option has been encountered, the version information field is changed to the following form in the NEWSOURCE file if the NEW option is TRUE, and in the LINE file if the LIST option is TRUE:

<release number>. <cycle number>. <version number>

The version number specified on the first VERSION option record replaces the version number on any subsequent VERSION option records. The <patch number> found in the version information field of the record being updated is retained. If the NEW option is TRUE, the version number in the NEWSOURCE file is updated. If the LIST option is TRUE, the version number in the LINE file is updated.

**Syntax**

```

<version option>
— VERSION —<release number>— . —<cycle number>— . —————>
-><patch number>—————|

<release number>
  ┌───┐
  │ /2\-<digit> │—————|

<cycle number>
  ┌───┐
  │ /3\-<digit> │—————|

<patch number>
  ┌───┐
  │ /3\-<digit> │—————|
  
```

**Explanation**

- <release number >  
Specifies a two-digit integer that updates the value of the release number.
- <cycle number >  
Specifies a three-digit integer that updates the value of the cycle number.
- <patch number >  
Specifies a three-digit integer that updates the value of the patch number.





# Appendix A

## SORT Language Keywords

The words listed in this appendix are recognized by the SORT compiler and fall into the following three categories:

- Category 1            Words that begin a SORT language statement. (See Table A-1 below.)
- Category 2            Words that appear within a SORT language statement to select various optional actions of that statement. (See Table A-2 below.)
- Category 3            File attribute names and attribute value mnemonics that are not listed in this manual. (For complete information regarding the words included in this category, refer to the *File Attributes Reference Manual*.)

Table A-1 alphabetically lists Category 1 words, their synonyms, and their compatibility with B 1000 Series Sort programs. Table A-2 alphabetically lists Category 2 words, their synonyms, and their compatibility with B 1000 Series Sort programs.

Table A-1. Category 1 Keywords

Keyword	Synonym or Synonyms	B 1000 Compatibility
BIAS		Yes
DELETE		No
DISKANDTAPE	ITD	No
DISKSORT		No
FILE		No
INCLUDE		No
KEY	KEYS, FIELD, FIELDS	No
MEMORY	ME	No
MERGE		No
NOPRINT		Yes
PARITY DISCARD		No
RECORDS		No
STABLE		No
SYNTAX		Yes
TAGSEARCH		No

continued

## **SORT Language Keywords**

---

**Table A-1. Category 1 Keywords (cont.)**

<b>Keyword</b>	<b>Synonym or Synonyms</b>	<b>B 1000 Compatibility</b>
TAGSORT		No
TAPESORT		No
TIME		Yes
TIMING		Yes
WORKFAMILY		No
WORKPACK1		Yes
WORKPACK2		Yes
WORKSIZE		No

**Table A-2. Category 2 Keywords**

<b>Keyword</b>	<b>Synonym or Synonyms</b>	<b>B 1000 Compatibility</b>
ALL		No
ALPHA		No
AND		No
ASCENDING	A	No
BIT		No
BYTE	BYTES	No
CARD	CARDS	Yes
CHARACTER		No
COMP		No
DECIMAL		No
DEFAULT		Yes
DESCENDING	D	No
DIGIT	DIGITS	No
DISK		Yes
DISPLAY		No
DOUBLE		No
EQL		No
EVEN	E	Yes

continued

Table A-2. Category 2 Keywords (cont.)

Keyword	Synonym or Synonyms	B 1000 Compatibility
FIRST		No
GEQ		No
GTR		No
HEX		No
IN		No
INTEGER		No
IS		No
LAST		No
LEADING	LEFT	No
LEQ		No
LSS		No
MULTI		Yes
NC		No
NEQ		No
NO COLLATE		No
NOT		No
NUMERIC		No
ODD	O	Yes
OR		No
OUT		No
PACKED		No
PAPER		Yes
PRINTER		Yes
PURGE		No
REAL		No
RELEASE		No
REMAPS		No
REMOVE		No
RSA		Yes
RSN		Yes

continued

**Table A-2. Category 2 Keywords (cont.)**

<b>Keyword</b>	<b>Synonym or Synonyms</b>	<b>B 1000 Compatibility</b>
SA		Yes
SAVE		Yes
SEPARATE		No
SIGN		No
SN		Yes
TAPE		Yes
TRAILING	RIGHT	No
UA		Yes
UN		Yes
V		Yes
VARIABLE		Yes
WORD	WORDS	No
ZONE		No

# Appendix B

## SORT Compiler Files

The SORT compiler uses the files listed in Table B-1. The table lists the files with their respective declarations and explains the function of each file.

**Table B-1. SORT Compiler Files**

File Name	File Declaration	Explanation
CARD	CARD (KIND = READER, FILETYPE = 8);	Used for SORT statement input.
SOURCE	SOURCE (KIND = PACK, FAMILYNAME = "DISK.", FILETYPE = 8, MYUSE = IN);	Used as the secondary source file for statements when the MERGE option value is TRUE in the CARD file.
NEWSOURCE	NEWSOURCE (KIND = PACK, FAMILYNAME = "DISK.", AREAS = 60, AREASIZE = 1008, FILEKIND = SORTSYMBOL, MAXRECSIZE = 15, BLOCKSIZE = 420, SAVEFACTOR = 999);	Used to create a new symbolic file when the NEW option value is TRUE.
LINE	LINE (KIND = PRINTER);	Used to print SORT specifications and error messages when the LIST option value is TRUE.
ERRORFILE	ERRORFILE (KIND = PRINTER, NEWFILE = TRUE, MAXRECSIZE = 14); ERRORS (KIND = PRINTER, NEWFILE = TRUE, MAXRECSIZE = 14);	Used to print error messages when the ERRORLIST option value is TRUE. If the SORT compiler was invoked from CANDE the file name ERRORS is used. Also, if the SORT compiler was invoked from CANDE, the KIND of this file is set to REMOTE; otherwise, the KIND is PRINTER.

continued

**Table B-1. SORT Compiler Files (cont.)**

<b>File Name</b>	<b>File Declaration</b>	<b>Explanation</b>
CODE	CODE (KIND = PACK, FAMILYNAME = "DISK.", MYUSE = OUT, AREAS = 40, AREASIZE = 504, FILEKIND = SORTCODE, FLEXIBLE, BUFFERS = 2, MAXRECSIZE = 30, BLOCKSIZE = 270, SAVEFACTOR = 999);	The resulting object code file from the SORT compiler. This file is produced only if the compilation is error-free.

# Appendix C

## SORT Program Files

Sort programs are compiled to use some or all of the files shown in Table C-1.

Table C-1. Sort Program Files

File Name	File Declaration	Explanation
IN	IN (KIND = PACK, FAMILYNAME = "DISK.", FILETYPE = 8, NEWFILE = FALSE);	Identifies the first input file to be sorted or merged.
IN2 through IN99	INn (KIND = PACK, FAMILYNAME = "DISK.", FILETYPE = 8, NEWFILE = FALSE);	Identifies the second through ninety-ninth input files.
OUT	OUT (KIND = PACK, FAMILYNAME = "DISK.", NEWFILE = TRUE, SAVEFACTOR = 99);	Identifies the sorted or merged output file.
DISKC	DISKC (KIND = PACK, FAMILYNAME = "DISK.", TITLE = "SORT/DISKC");	Used by the system SORT procedure as its control file. Changing any attribute other than TITLE or FAMILYNAME can cause the sort to fail.
DISKF	DISKF (KIND = PACK, FAMILYNAME = "DISK.", TITLE = "SORT/DISKF");	Used by the system SORT procedure as its work file. Changing any attribute other than TITLE or FAMILYNAME can cause the sort to fail.

**Note:** Using *FILETYPE = 8* indicates that certain attribute values are to be obtained from the permanent file even if these attributes are specified in the Sort program (for example, *MAXRECSIZE*). To change the value of these attributes, change the *FILETYPE* attribute. For more information concerning the attributes affected by *FILETYPE = 8*, refer to the File Attributes Reference Manual.

The file attributes for the files IN, IN2 through IN99, and OUT can be left as the default values specified in the appropriate SORT statements, or can be changed by file attribute assignments at the time the Sort program is initiated from WFL or CANDE.



## **SORT Program Files**

---

If an explicit value is not declared, the OUT file assumes the values associated with the IN file for the following file attributes:

AREASIZE	BLOCKSIZE	EXTMODE
FAMILYNAME	FILEKIND	FILETYPE
INTMODE	MAXRECSIZE	SECURITYGUARD
SECURITYTYPE	SECURITYUSE	UNITS

If both WFL file equations and internal file specifications are present, the WFL file equations override the internal file specifications.

# Appendix D

## B 1000 SORT Conversion

The A Series SORT compiler accepts most B 1000 Sort syntax, thus easing the migration of B 1000 Series system users to A Series systems. This appendix explains the differences between the SORT compiler and the B 1000 Sort program. The additional syntax accepted by the compiler for conversion is also described.

### B 1000 Statements Accepted but Ignored

The following B 1000 SORT statements are accepted but ignored by the A Series SORT compiler. A warning is issued by the compiler when the following statements are used:

- BIAS
- TIME
- TIMING

### B 1000 Statements Not Supported by the SORT Compiler

A syntax error occurs when the following B 1000 SORT statements are used:

COLLATE	DUPCHECK
INPLACE	OVERRIDE
TAGCOBOL	TAGRPG
ZIP	

The B 1000 TEACH statement has been replaced by the \$REFORMAT compiler control option. The B 1000 NOPRINT statement has been replaced by the \$LIST compiler control option.

The B 1000 handling of embedded comments, which are unrecognized words the Sort program ignores, is not supported by the A Series SORT compiler. A Series SORT syntax requires that comments follow a percent sign (%) or a colon (:) if they are used on an input record.

### B 1000 File Statement

The B 1000 FILE statement is composed of the reserved word FILE followed by 1 to 99 input files and one output file. The A Series SORT compiler allows up to 99 input files to be sorted and up to eight files to be merged. Comparatively, the B 1000 Sort program allows up to 16 input files to be sorted.

## B 1000 SORT Conversion

**Note:** If the <B 1000 file name> is IN or OUT, the left parenthesis and the device (DISK, CARD, CARDS, PAPER, or TAPE) must be on the same input record; otherwise, A Series SORT syntax is assumed.

### Syntax

<B 1000 file statement>

— FILE — /99\—<B 1000 file input part>—<B 1000 file output part>—

## B 1000 File Input Part

The <B 1000 file input part> describes an input file.

### Syntax

<B 1000 file input part>

— IN —<B 1000 file name>— ( ————— )

→ DISK — ( —<rpa>— ) —<rsz>—<r.b>— PURGE — MULTI —

→ CARD — DEFAULT —<rsz>—<r.b>—

→ CARDS —<rsz>—<r.b>—

→ PAPER —

→ TAPE — (ODD) —

→ (O) —

→ (EVEN) —

→ (E) —

→ V( —<maximum block size>— ) —

→ VARIABLE —

<rpa>

—<integer>—

<rsz>

—<integer>—

<r.b>

—<integer>—

<maximum block size>

—<integer>—

### Explanation

#### DISK

Specifies that the input file is a disk or disk pack file.

CARD, CARDS

Specifies that the input file is a card reader file.

PAPER

Specifies that the input file is a paper tape reader file.

TAPE

Specifies that the input file is a tape file. If the parity specifier is included after the keyword, TAPE, then 7-track tape is implicitly specified. A parity specifier is enclosed in parentheses. ODD, or O, specifies odd parity, binary mode. EVEN, or E, specifies even parity with Burroughs Common Language (BCL) translation. If even-parity, 7-track magnetic tape is specified, the SORT compiler issues a warning about the incompatibility of BCL characters with EBCDIC-only machines.

< rpa >

Specifies the number of records per area. This number must be enclosed in parentheses. An < rpa > specification is used when the input file is located on disk and the DEFAULT option is not used.

< rsz >

Specifies the actual record size, in bytes, of the records to be sorted. Unless the DEFAULT option is specified, the < rsz > is a required entry.

< r.b >

Specifies the number of logical records in a block. By default, < r.b > is 1.

DEFAULT

Specifies that the B 1000 Sort program is to obtain the input file specifications from the disk file header, if the input file is on disk. The input file specifications include records per area (< rpa >), record size (< rsz >), and records per block (< r.b >).

PURGE

Specifies that the input file is to be removed from the disk directory if the input file is a disk file. However, if the input file is a magnetic tape file, the tape is purged. This option can affect only disk and tape input files.

MULTI

Specifies that the file occupies more than one disk pack. The SORT compiler does not recognize this option and ignores it.

## B 1000 SORT Conversion

### VARIABLE

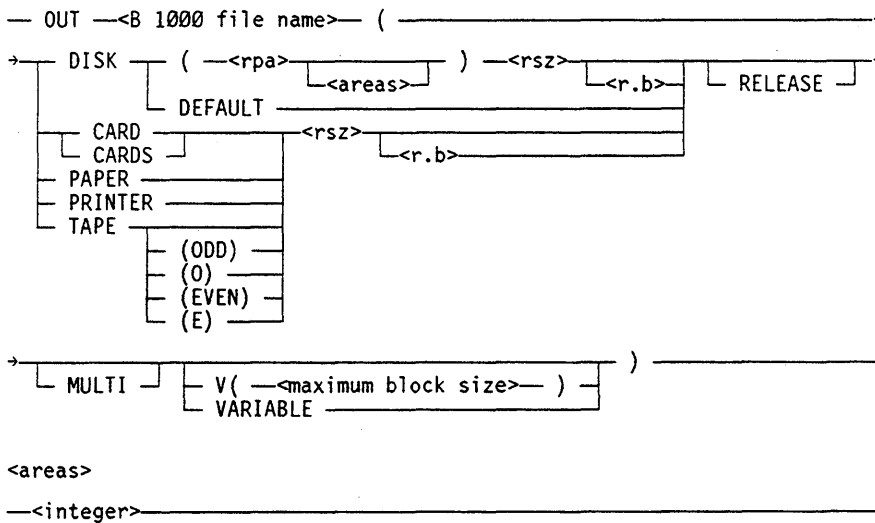
Specifies that the file contains variable-length records. The SORT compiler recognizes the B 1000 syntax for variable-length records, but issues a syntax error because variable-length record capability is not implemented.

## B 1000 File Output Part

The <B 1000 file output part> describes the one required output file. The elements of the <B 1000 file output part> have the same functions for the output file as the elements of the <B 1000 file input part> have for the input file.

### Syntax

<B 1000 file output part>



### Explanation

#### DISK

Specifies that the output file is a disk file.

#### CARD, CARDS

Specifies that the output file is a card punch file.

#### PAPER

Specifies that the output file is a paper tape punch file.

#### TAPE

Specifies that the output file is a tape file.

**DEFAULT**

Causes the records per area (<rpa>), record size (<rsz>), and records per block (<r.b>) of the output file to be identical to those of the first input file.

<areas>

Specifies the maximum number of disk areas allowed in the output file.

**RELEASE**

Specifies that the tape is to be closed with release. This option is valid only for tape files.

**VARIABLE**

Specifies that the file contains variable-length records. The SORT compiler recognizes the B 1000 syntax for variable-length records, but issues a syntax error message because variable-length record capability is not implemented.

**Example**

```
FILE IN X (TAPE 80 1)
      OUT X/Y/Z (DISK (900 25) 80 9)
```

**B 1000 File Name**

Each level of a <B 1000 file name> can consist of any character. The characters comma (,), blank space ( ), asterisk (\*), slash (/), and quotation mark (") each have a particular meaning. The quote character can be represented by a pair of quotation marks (" ") and each level of the <B 1000 file name> can be enclosed in quotes. If the <B 1000 file name> is enclosed in quotes, the blank and slash lose their associated meanings; the asterisk, the left parenthesis, and the right parenthesis retain their associated meanings.

Two forms of the <B 1000 file name> cannot be completely handled on A Series systems:

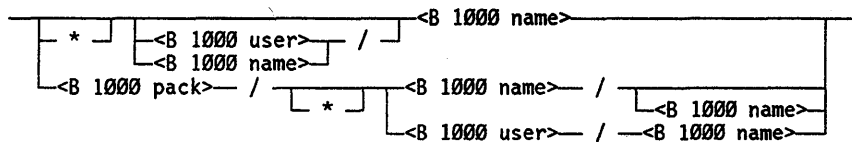
- On B 1000 Series systems, a usercode enclosed in parentheses and preceded by an asterisk indicates that no family substitution is to be performed and that the file is found on the system resource disk.
- The normal usercode form, which is not preceded by an asterisk, indicates that the referenced disk file is to be found on the default family for the usercode that owns the file.

These actions cannot be simulated by the SORT compiler. Instead, the SORT compiler translates the <B 1000 file name> forms as best it can and issues a warning indicating that the family substitution action is different.

# B 1000 SORT Conversion

## Syntax

<B 1000 file name>



## Explanation

< B 1000 name >

Specifies the file name. This specification can contain no more than ten EBCDIC alphanumeric characters, not counting the enclosing quotes. Any embedded quotation characters (") are counted as one character. An asterisk (\*) can be used along with a maximum of nine additional characters, which indicates that a usercode is not applied to the file name.

< B 1000 user >

Specifies the usercode under which the file resides. This specification can include no more than eight EBCDIC alphanumeric characters enclosed in parentheses. If an asterisk (\*) precedes the < B 1000 user > specification, a family substitution does not take place, and only seven alphanumeric characters can be used.

< B 1000 pack >

Specifies the pack. This specification can include up to ten EBCDIC alphanumeric characters.

## Examples

Table D-1 shows example B 1000 file names and their compatible TITLE attribute values on A Series systems.

Table D-1. Equating B 1000 File Names and A Series TITLE Attributes

B 1000 File Name	TITLE
A	A ON DISK
A/B	A/B ON DISK
A/B/	B ON A
A/B/C	B/C ON A
A(B)/C	(B)C ON A
*A	*A ON DISK

continued

Table D-1. Equating B 1000 File Names and A Series TITLE Attributes (cont.)

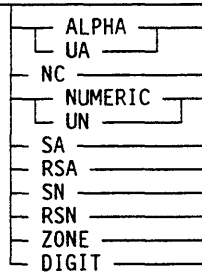
B 1000 File Name	TITLE
*A/B	*A/B ON DISK
A/*B/	*B ON A
A*B/C	*B/C ON A
A*(B)/C	(B)C ON A
(A)/B	(A)B ON DISK
*(A)/B	(A)B ON DISK

## B 1000 Data Type

The < B 1000 data type > defines the format of a data item.

### Syntax

<B 1000 data type>



### Explanation

#### ALPHA, UA, NC

Indicates that the data is alphanumeric and that the key location in the input record is counted in bytes from the beginning of the record. The comparison is based on the character value of the field. ALPHA is the default data type for B 1000 Series systems. This type is equivalent to the ALPHA <type> of the SORT compiler.

#### NUMERIC, UN

Indicates that the data is 4-bit numeric and that the relative position of the field is counted in 4-bit units. The comparison is based on the numeric value of the field. This type is equivalent to the PACKED <type> of the SORT compiler.



### SA

Indicates that the data is 8-bit alphanumeric and that the key or data field contains a sign field. The minus sign (-) is represented by a hexadecimal D in the most significant four bits, or zone portion, of the first byte of the field. Any other value in the sign represents a plus sign (+). This type is similar to, but not the same as, ALPHA NUMERIC SIGN LEFT.

An SA field uses all bits of all characters in the field for the key comparison. In contrast, an ALPHA NUMERIC SIGN LEFT field uses only the digit portion of each character and the zone portion of the first character for the key comparison. The same literals and literal conversions used for ALPHA NUMERIC fields are used for SA fields. (Refer to the explanations of literal and literal conversions in the "Sort Program Structure" section.)

### SN

Indicates that the data is 4-bit numeric and that the key or data field contains a sign in the first digit of the field. The minus sign (-) is represented by a hexadecimal D. Any other value represents a plus sign (+). SN is the B 1000 equivalent of the PACKED SIGN LEFT data type.

### RSA

Specifies that the data is signed 8-bit alphanumeric, and that the minus sign (-) is represented by a hexadecimal D. The sign is located in the first four bits, or zone portion, of the rightmost byte of the key. Any other value represents a plus sign (+). This data type is similar to, but not the same as, ALPHA NUMERIC SIGN RIGHT.

An RSA field uses all bits except the zone portion of the first character for key comparison; an ALPHA NUMERIC SIGN RIGHT field uses only the digit portion of each character and the zone portion of the first character for key comparisons. The same literals and literal conversions used for ALPHA NUMERIC fields are used for RSA fields. (Refer to the explanations of literal and literal conversions in the "Sort Program Structure" section.)

### RSN

Indicates that the data is signed 4-bit numeric, and that the minus sign (-) is represented by a hexadecimal D in the rightmost digit of the key. Any other value represents a plus sign (+). RSN is equivalent to the PACKED SIGN RIGHT data type.

### ZONE

Equivalent to the A Series ZONE data type.

### DIGIT

Equivalent to the A Series DIGIT data type.

## B 1000 INCLUDE and DELETE Statement

The IN option of the < B 1000 include and delete statement > is not supported on A Series systems; however, SYSTEM/DUMPALL provides similar capabilities.

The B 1000 Sort program specifies that multiple AND and OR operations are evaluated from left to right. The SORT compiler, like the B 1000 Sort program, specifies that the order of precedence is NOT, AND, and OR. Some B 1000 Sort programs may need to be modified by adding parentheses so that the desired evaluation of Boolean conditions is obtained.

With the B 1000 Sort program, if a DELETE is followed by an INCLUDE and both statements select the same record for inclusion and deletion, the record is retained. The SORT compiler, however, uses the same rule as the B 1000 Sort program: each successive INCLUDE or DELETE creates further subsets of the input records.

## B 1000 MEMORY Statement

The < B 1000 memory statement > specifies memory in units of bytes. The SORT language < memory statement > specifies memory in units of 48-bit words. The number specified in a < B 1000 memory statement > should remain unchanged in a SORT language < memory statement >. A sort that runs on B 1000 Series systems in 15,000 bytes will run on A Series systems in 15,000 words.

## B 1000 MERGE Statement

A < merge statement > is assumed by the B 1000 Sort program if more than one input file is specified in the absence of other SORT processing statements. The SORT compiler accepts the absence of the < merge statement > or < disksort statement >, but does issue a warning message.

## B 1000 NOPRINT Statement

The < B 1000 noprint statement > performs the same function as resetting the LIST compiler control option. For example, the statements NOPRINT and \$ RESET LIST have the same effect.

### Syntax

<B 1000 noprint statement>

— NOPRINT —————|

## B 1000 RECORDS Statement

The SORT language < records statement > can be specified on B 1000 Series systems using the following syntax.

### Syntax

<B 1000 records statement>

—<record estimate>— RECORDS —————|

## B 1000 SORT Statement

The B 1000 Sort program performs a disk sort in the absence of a < tapesort statement > or a < B 1000 tapesort statement > when only one input file is specified. The SORT compiler generates a < disksort statement > if only one input file exists and a < merge statement >, < disksort statement >, < diskandtape statement >, < memoryonly statement >, or < tapesort statement > is not specified. A warning message is generated by the SORT compiler.

## B 1000 SYNTAX Statement

The < B 1000 syntax statement > forces the SORT compiler to suppress production and output of the CODE file. The effect of the command is the same as a syntax only compilation.

### Syntax

<B 1000 syntax statement>

— SYNTAX —————|

## B 1000 TAGSORT Statement

The B 1000 Sort program creates 4-byte, or 8-digit, index records if the < B 1000 tagsort statement > is specified. Conversely, the SORT compiler creates this type of index record only for a PACKED SORT type with a < length > of 8. A one-word record is created by default.

The Report Program Generator (RPG) ADDRROUT files supported by the A Series RPG compiler consist of one-word records. Therefore, ADDRROUT files taken directly from a B 1000 Series system are incompatible with A Series RPG. However, Sort programs taken from B 1000 Series systems and run on A Series systems create ADDRROUT files that are compatible with A Series RPG.

## B 1000 TAPESORT Statement

The SORT language <tapesort statement> can be specified on B 1000 Series systems using the syntax illustrated in the following diagram.

### Syntax

<B 1000 tapesort statement>

—<work tapes>— TAPESORT —————|

## B 1000 WORKPACK1 and WORKPACK2 Statements

The SORT compiler recognizes and accepts the B 1000 WORKPACK1 and WORKPACK2 statements. The last <B 1000 workpack statement> encountered in an A Series Sort program is changed to a <workfamily statement>, and a warning is issued.

### Syntax

<B 1000 workpack statement>

— /1\— WORKPACK1 —<B 1000 pack>—————|  
 — /1\— WORKPACK2 —|

### Explanation

WORKPACK1, WORKPACK2

Specifies that a particular disk pack family is to be used by the Sort program work files.

<B 1000 pack>

Specifies the disk pack to be used.



# Appendix E

## Using GSORT to Code and Run Sort Programs

A GSORT program is a Sort program where the GSORT option is set to TRUE. Once the option is set, it cannot be reset.

### How GSORT Programs Are Similar to Sort Programs

Every Sort program, whether the GSORT option is set or not, identifies the records to be sorted and the key fields for sorting. The following example shows how a Sort and GSORT program can use the same input file to produce the same output file.

#### Example: Sort and GSORT Programs Producing the Same Result

For this example, assume that your input file consists of the following five records:

Input Records

1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	123456789012
first name	last name	customer number	item number	quantity ordered	status	of order
Henry	Molloy		222	4444		333W
James	Meehan		111	3333		222S
Nancy	Jones		333	5555		222W
Don	Meehan		555	5555		111S
Iris	Johnson		444	5555		111W

Now assume you want to sort the records alphabetically, by last name. You want the entire input record, as is, to be in your output file; you want your output file to have the following order and format:

Iris	Johnson		444	5555		111W
Nancy	Jones		333	5555		222W
James	Meehan		111	3333		222S
Don	Meehan		555	5555		111S
Henry	Molloy		222	4444		333W

The following Sort program produces this result:

```
DISKSORT
FILE IN(TITLE ="GSORT/MAN/WORKFLOW")
FILE OUT
STABLE
KEY(10 29)
```

The following GSORT program produces the same result:

```
00000$SET GSORT
00200HSORTRE 20A      0X 60      REGULAR SORT, FIFO EQUAL KEY
00300FNC 10 29      KEY FLD: LAST NAME 10..29
00350FDC 1 60      DATA FLD: ALL OF INPUT RECORD
```

*Note: Unless the sort process has specific instructions for handling identical (equal) key fields, each sort process can have a different outcome. In these examples the instructions state equal key fields should be handled on a first in, first out (FIFO) basis; therefore, since the record for James Meehan is read first, it comes before the record for Don Meehan in the output file.*

## How GSORT Programs Differ from Sort Programs

There are four major differences between GSORT programs and Sort programs:

- With GSORT you can remap output records.
- With GSORT you can use alternate collating sequences.
- With GSORT you must use a column-oriented syntax
- With GSORT you must specify the names of the input and output files at run time using the standard file-equation syntax.

## Creating Output Records by Remapping Input Records

For Sort programs where the GSORT option is not set, the input and output records have the same fields and format.

However, when GSORT is set, you can remap the input record to create output records where fields are excluded or repositioned. For instance, using the sample input record, you could create an output record that contained only the customer number, item number, quantity ordered, and status of order; the first and last name fields could be excluded. And, you could rearrange the remaining fields so that the status was first, the item number second, the quantity third, and the customer number last.

### Input Record

1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
first	last name	customer	item	quantity	status	
name		number	number	ordered	of order	

### Output Record

1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
status of	item	quantity	customer	----	not used	----
of order	number	ordered	number			

Remapping does not change the content of the fields, nor does it change the input record.

Additional information on creating output records can be found under “Determining the Content of the Output File” in this appendix. The actual syntax used to select data and key fields and to remap records is covered in the detailed discussions of the GSORT specifications in this appendix.

## Using Alternate Collating Sequences

Collating sequences are logical sequences used to sort items of data into a particular order. A collating sequence is used to compare characters in the key fields in the input records to determine whether one character is equal to, greater than, or less than another character.

By default, output is ordered according to the standard EBCDIC collating sequence. (See Appendix F, “Using the Standard Collating Sequence” for details of the EBCDIC collating sequence.) However, when the GSORT option is set, you can change the collating sequence for all or selected characters, the entire key field, or specified key fields.

With an alternate collating sequence you can insert a character between two existing characters, (such as a dollar sign between the letters A and B), you can take a character out of the usual sequence (such as BAC instead of ABC), or you can exchange characters (put B where W is and W where B is). You can also use the system default internationalization collating sequence to process alphanumeric data.

For more detailed information see “Coding Header Information,” “Coding ALTSEQ Statement,” and “Coding Field Selection Information.”



### Coding GSORT Programs

Instead of using SORT statements, a GSORT program is coded using GSORT specifications and statements. The GSORT syntax is similar to RPG II syntax and is compatible with the industry-standard GSORT language.

- The sort specifications, including key fields, are entered through Header, Record Selection, and Field Selection records.
- An alternate collating sequence statement (ALTSEQ) is available.
- No FILE statement can appear within the GSORT program. Instead, the names of the input and output files are specified at run time.
- All GSORT specifications and statements use fixed-format, 72-column entries.
- Only one specification or statement can be coded on each program record.
- There are no semicolons (;) between specifications and statements.
- Comments are not preceded by a percent sign (%) or a colon (:). Instead, they are identified as comments by the columns they appear in (for specifications, columns 44 through 72).
- Sequence numbering is in columns 1 through 5, not columns 73 through 80.

The GSORT specifications and statement allow you to identify the operations to be performed and the records and fields to be used in the sorting process. The valid GSORT specifications and statement are

- A required Header specification that describes the type of sort you want to run
- An optional ALTSEQ statement that describes any alternate collating sequence
- One or more optional Record Selection specifications that describe the input file records you want to include in or omit from the sort
- One or more required Field Selection specifications that indicate how you want the input records to be sorted and identify the data and key fields you want written into the output file

In a GSORT program, the specifications and statement must be entered in a specific order. The Header specification must precede any ALTSEQ statement. ALTSEQ statements must precede any Record Selection specifications. Record Selection specifications must precede any Field Selection specifications. However, you might not have to complete all the specifications or statements. For example, if you want to sort all the records in a file and they all are of the same type, you do not need Record Selection specifications. You need only Header and Field Selection specifications. Or, if you want to use the standard collating sequence, do not code an ALTSEQ statement.

In a GSORT program, you can create subsets of records. These subsets are known as types. For example, you might define one type as all payroll records with a D in column 7, another type as all payroll records with a Y in column 7, and a third type as all payroll records with a 3 in column 4 and a T in column 20. All record types are user-determined.

Refer to “Coding Header Information,” “Coding ALTSEQ Statements,” “Coding Record Selection Information,” and “Coding Field Selection Information” for more details.

### Executing a GSORT Program

When you execute a GSORT job, the program first compiles your sort specifications. Each specification is processed as it is encountered in the program. If the Sort compiler detects errors, error messages are displayed and can be printed.

For an explanation of run-time errors, see “GSORT System Error Messages.”

To run a GSORT job, after compiling the program use the *CANDE RUN* command. Default input and output file names are used unless other file names are entered using a FILE equation. (No FILE statement appears within the program.)

A maximum of eight files can be input to a GSORT program. Each named input file must be associated with a unique number. The valid entries are

- INPUT or INPUT1 (but not both)
- INPUT2
- INPUT3
- INPUT4
- INPUT5
- INPUT6
- INPUT7
- INPUT8

If no input file name is specified, a default input file named INPUT is used.

You can enter the input files in any sequence; regardless of the order of the FILE equation, the input files are processed serially. First, the file with the lowest number is processed fully, then the file with the next lowest number, and so on until each input file has been processed individually. For example, a job could have the file names entered in the order of INPUT2, INPUT7, and INPUT3, but the order of the actual sort would be INPUT2, INPUT3, and INPUT7.

As with Sort programs, by naming more than one input file, you can merge files; that is, you can combine the records from two or more input files into one output file. However, with GSORT programs, the records in each input file can be in a different order and the input data can be rearranged, reformatted, or dropped in the output record.

Only one output file is produced. If no output file name is specified, the default file name is OUTPUT.

### Example: Specifying No Input or Output File

In the following example, the GSORT program ASCENDING is executed. Because neither an input nor an output file is specified, the defaults are used. The system assumes there is one input file, named INPUT. The output file is named OUTPUT.

```
RUN ASCENDING
```

### Example: Specifying an Output File

Shown below, the GSORT program ACCTNO is executed. The system assumes there is one input file named INPUT. The output file is named ACCTOUT.

```
RUN ACCTNO; FILE OUTPUT (TITLE=ACCTOUT)
```

### Example: Specifying Four Input Files and an Output File

The GSORT program PAYROLL is executed in the next example. The input files REGPAYROLL, SPCLPAYROLL, HOLPAYROLL, and VACAPAYROLL are specified. The output file is named PAYROLLOUT.

```
RUN PAYROLL;  
FILE INPUT (TITLE=REGPAYROLL);  
FILE INPUT3 (TITLE=HOLPAYROLL ON PACK);  
FILE INPUT2 (TITLE=SPCLPAYROLL ON PACK);  
FILE INPUT4 (TITLE=VACAPAYROLL ON PACK);  
FILE OUTPUT (TITLE=PAYROLLOUT)
```

## Determining the Content of the Output File

Through the GSORT specifications and statement, you identify the type of sorted output you want and the records and the key fields to sort on. The sorted output file can contain

- Parts or all of the records in the input file. This is known as a regularly sorted output file or SORTR.
- Summarized fields for each unique key field in the input file. This is known as an accumulated totals output file or SORTRS.
- The relative record numbers of records in the input file. This is known as a record numbers output file or SORTA. (It is also known as a tag sort or an ADDROUT sort.)

### Criteria for Regularly Sorted Output (SORTR)

In regularly sorted output (SORTR), a single output record containing the data you selected from the input record is created for each input record and written to the output file in the order you specified. SORTR is the most common type of output file.

SORTR output files can consist of records containing the following information:

- Both key and data fields
- Key fields only
- Data fields only

You must provide the key field identification.

You can also remap the output records. You must identify each field by its starting and ending positions. The sort specifications assign each field an order; this order determines how the input fields are to be placed in the output record.

You can arrange the data in regularly sorted files from the lowest value to the highest value (ascending order), or from the highest value to the lowest value (descending order).

### Criteria for Sorted Files with Accumulated Totals (SORTRS)

To add data from particular fields and produce a file containing the totals, use output files with accumulated totals or summary data (SORTRS).

Output files of summary data can contain any of the following kinds of records:

- Data fields with accumulated totals only
- Both data fields with accumulated totals and key fields
- Data fields with accumulated totals, key fields, and other data fields

### Criteria for Files of Record Numbers (SORTA)

You can use a GSORT program to sort records selected from a single file and produce a record number file that reflects the new order. The record number file contains 3-byte relative record numbers only of some or all the records in the input file. Relative record numbers identify the physical location of records in the original input file. The relative record number of the first record in an input file is always zero (00 00 00). The data from the input file are not written to the output file.

The SORTA output can be used by a Report Program Generator (RPG) program as an ADDROUT file or by programs that process files of relative record numbers. A single output record is created for each selected input record. Only one input file at a time can be used to produce this type of output.

You must provide the GSORT program with the following information:

- Record identification information. Different record types are user-determined and can be used in the same sort.
- Key fields.

Any Field Selection specifications you enter are ignored. No data fields are output.

### Selecting the Fields in an Output Record

Use the Field Selection specification to identify the fields in the input records that are to be written into the output records. The order in which they are specified in the program records determines the order in which they are written to the output record.

A key field can be either normal, opposite, or forced. A normal key field is sorted in the order specified in the Header record. An opposite key field is sorted in the opposite order. A forced key field forces the record to be placed in a particular sequence, regardless of any other specified order.

A data field can be normal data, summary data, or forced data. A normal data field is included in the output record, but not used in the sorting process. A summary data field holds accumulated totals. A forced data field forces a character constant at selected locations in the output data.

All data fields noted in a Field Selection record are included in the output file. To specify that key fields should be written into the output file as well as data fields:

- Enter the field as a key field in a Field Selection record and leave the Output Option column of the Header record blank.
- For normal key fields with packed or zone data, opposite key fields, or an alternate collating sequence, the key fields must be described twice in different Field Selection records: once as key fields and again as data fields. Place an X in the Output Option column of the Header record.
- If you use equal key field ordering, describe the field twice: once as a key field and again as a data field. Place an X in the Output Option column of the Header record. (Otherwise, the output file includes the 3-byte relative record number GSORT uses to order the records that have equal key fields.)

To have the key field appear in the output record, you must describe it once as a key field and once as a data field. The GSORT program uses the key field description to transform the contents of key fields for sorting efficiency. (The transformed key fields might not be the same as the original input fields. If they are not the same, the description is invalid as a data field in the output record.)

Unless you write the key field into the output file, the output record length includes only the data fields from the input file.

See "Coding Field Selection Information" for a more detailed explanation of the types of key and data fields.

### Coding Header Information

The Header specification is used to state the type of sort and some of the sort parameters. Each GSORT job must have one, and only one, Header specification. The Header specification must come before any other specification or statement.



**Table E-1. Header Specification Entry Table (cont.)**

Columns	Valid Entries	Purpose of Entry
		Display: Action and error messages.
	1	Print: Program status messages.
		Display: Action and error messages.
	2	Print: Action messages only.
		Display: Error messages.
	3	Display: Error messages.
28	Blank	Indicates key field data are written in the output file. Valid for SORTR and SORTRS.
	X	Indicates key field data are not to be written into the output file. Valid for SORTR and SORTRS.
29-32	1-4096	Indicates the length of the output record. Required for SORTR and SORTRS.
33-35		Not used.
36	Blank	Specifies that a warning message be printed or displayed when no records are selected for sorting.
	N	Specifies that a warning message not be printed or displayed when no records are selected for sorting.
37-43		Not used.
44-72	Any characters	Comments.

### Sequence Number (Columns 1 through 5)

Sequence numbers are optional. If entered, they must be in ascending order. If the numbers are not in ascending order, when the specifications are being processed an error message is placed next to the statement that is out of sequence.

Any item in these columns that is not a number is ignored by the program.

### Header Specification (Column 6)

An H must appear in column 6 to identify a Header specification.

## Type of Job (Columns 7 through 12)

Three types of sort output files can be produced with GSORT. Use columns 7 through 12 to specify the type of GSORT output you want.

Enter SORTR for regularly sorted output.

Enter SORTRS for sorted output with accumulated totals, or a summary Sort.

Enter SORTA for sorted output in the form of relative record numbers.

## Identical, or Equal, Key Fields (Column 12)

When key fields in different input records have identical values, it is not possible to predict the order in which these records will be written into the output file unless you specifically state the order.

For SORTR or SORTA jobs, to maintain records with equal key fields in their original input sequence, place an E in column 12. Otherwise, for ascending sorts, records with equal key field values are placed in the output file in the order in which they appear in the input file. For descending sorts, records with equal key field values are placed in the output file in reverse order from their order in the input file. (See column 18 for an explanation of ascending and descending sorts.)

To merge two or more input files, use the equal key field ordering. When merging multiple input files, SORTA (ADDROUT sort) is not allowed.

## Key Field Lengths (Columns 13 through 17)

Enter the sum of all key field lengths, taken from columns 9 through 16 of the Field Selection specifications. (The total cannot exceed 256.) Right-justify the entry.

The following steps summarize the way to calculate key field lengths:

1. Calculate the total key field length for each record type defined, in the following way:

Add the lengths of all normal and opposite key fields defined for the record type (N or O in the Field Type column of the Field Selection specification).

Add 1 to the sum for each forced key field (F in the Field Type column and a blank in the Continuation column of the Field Selection specification). The additional character is needed to reserve space for the single character constant that is forced into the key field.

Add 3 to the sum if you are sorting identical key fields and you specify equal key field ordering (E in the Equal Fields column of the Header specification). The additional 3 characters are needed to reserve space for the 3-character record number used by the GSORT program.

2. Enter the largest sum of all the record types in the Key Field Length columns of the Header specification. Right-justify the entry.



### Record-Sorting Order (Column 18)

To indicate the order in which you want your input records sorted, enter either of the following:

Entry	Request
A	Ascending order by key field.
D	Descending order by key field.

Ascending order means that the record placed first in the output file contains the item with the lowest value (lowest to highest value). Descending order means that the record placed first in the output file contains the item with the highest value (highest to lowest value).

To specify how to sort records with equal values in key fields, use column 12.

### Columns 19 through 25

These columns are not used.

### Collating Sequence (Column 26)

Use this column to specify if you are using the standard collating sequence or an alternate collating sequence.

The available options for this column are

Entry	Meaning
Blank	Use the standard collating sequence.
N	Use the system default internationalization collating sequence for alphanumeric fields.
S	Use an alternate sequence on the entire key field.
F	Change the standard collating sequence for specified key fields only.

If you specify an alternate collating sequence, you must also provide ALTSEQ statements immediately after the Header specification.

If you put an F in column 26, it changes the standard collating sequence for normal and/or opposite key fields only.

Place an A in column 20 of the key field line on a Field Selection specification to change a normal or opposite key field. You must also supply ALTSEQ statements immediately following the Header specification.

## Print Option (Column 27)

This column indicates the types of messages that will be automatically printed for a job, as needed. The options for column 27 are

Entry	Meaning
0 or Blank	GSORT specifications Diagnostic messages Program status messages Action messages Displayed messages
1	Program status messages Action messages Displayed messages
2	Action messages Displayed messages
3	Displayed messages

When you are testing your GSORT job, use a 0 (zero) or a blank. After you have run the job successfully, use a 3 for displaying messages.

## Output Option (Column 28)

This entry indicates whether key field data should appear in the output file. (Valid only for SORTR and SORTRS output.) The options are as follows:

Entry	Request
Blank	Write key field data into the output file.
X	Do not write key field data into the output file.

## Output Record Length (Columns 29 through 32)

This entry specifies the length of records in the output file. The output records can be from 1 to 4096 characters long. Note that

- If you do not drop key fields, the output record length should include both the length of the key fields and the length of the specified data fields.
- If you drop key fields, the output record length includes only the length of the specified data fields.
- This entry is valid only for SORTR or SORTRS output.

- If you specify E (in column 12 of the Header specification) for equal fields, the key field length is increased by 3 bytes and the output record length is increased by 3 bytes because the key field is added to the output data fields.

### Columns 33 through 35

These columns are not used.

### Null Output (Column 36)

If the GSORT program reads all the input records and no records are selected for processing, the entry in this column determines whether an error message should be displayed.

Entry	Meaning
Blank	Display and print warning messages.
N	Do not display or print warning messages.

If a warning message is displayed while your GSORT job is running, at a terminal you can enter one of the following GSORT standard options using the `?<mix number> AX` command.

Entry	Request
0	Continue running the job without creating an output file.
3	Cancel the job without creating an output file.

### Columns 37 through 43

These columns are not used.

### Comments (Column 44 through 72)

These columns can be used to enter optional comments and to document the program. If you specify that the GSORT job print your specifications (the column 27 entry of the Header specification is zero or blank), the comments you include in these columns are printed.

### Coding ALTSEQ Statements

An alternate collating sequence changes the collating sequence for all or selected characters, for the entire key field, or for specified key fields. It never changes the collating sequence for data fields or forced key field characters.

An ALTSEQ statement, consisting of one or more records, defines the new collating sequence. In the statement, both the character you want to shift and the character you want to replace it with are identified by their hexadecimal values. Each record can

specify a maximum of 18 pairs of hexadecimal equivalents. A maximum of 96 pairs can be designated. Regardless of how you alter the sequence, you must specify every pair of characters to be changed.

**Notes:**

- *When you use packed or unpacked fields with an alternate collating sequence, a portion of the packed or unpacked characters could be translated and become invalid numbers.*
- *Defining an alternate collating sequence also requires an entry in column 26 of the Header specification, and, as needed, entries in column 20 of the Field Selection specifications.*

Table E-2 summarizes the column entries for an ALTSEQ statement. The entries are described more fully in the text that follows the table.

**Table E-2. ALTSEQ Statement Entry Table**

Columns	Valid Entries	Purpose of Entry
1-6	ALTSEQ	Identifies the record as part of the ALTSEQ statement.
1-2	**	Identifies the end of ALTSEQ statement.
7-8		Not used.
9-10	Hexadecimal equivalent	Hexadecimal equivalent of the character being taken out of its normal sequence.
11-12	Hexadecimal equivalent	Hexadecimal equivalent of the character being inserted into the sequence.
13-14	Hexadecimal equivalent	Hexadecimal equivalent of next character being taken out of its normal sequence.
15-16	Hexadecimal equivalent	Hexadecimal equivalent of next character being inserted into the sequence.
17-72	Hexadecimal equivalents	Pairs of hexadecimal equivalents (characters being taken out of sequence and characters being inserted into sequence).

**Name of Statement (Columns 1 through 6)**

The word ALTSEQ must appear in columns 1 through 6 to identify it as part of the ALTSEQ statement.

**Double Asterisks (Columns 1 and 2)**

Enter two asterisks (\*\*) to end the ALTSEQ statement.

### (Columns 7 and 8)

These columns are not used.

### Character Being Taken Out (Columns 9 and 10)

Enter the hexadecimal equivalent of the character you are taking out of its normal sequence.

### Character Being Inserted (Columns 11 and 12)

Enter the hexadecimal equivalent for the character being inserted into the sequence in place of the character noted in columns 9 through 10. The inserted character assumes the value of the character being taken out.

*Note: Continue entering the pairs of the characters you are taking out of normal sequence. Leave no spaces between sets of hexadecimal numbers. When you reach the end of one record line, you can continue on the next statement line.*

## Using Alternate Collating Sequences for an Entire Key Field

For an alternate collating sequence on the entire key, place an S in column 26 of the Header specification and enter an ALTSEQ statement. During processing, the GSORT program first changes the entire input record to the specified alternate sequence of characters. Doing so establishes the new collating sequence. Any include or omit record type checks are run against the alternate sequence data.

The ALTSEQ statement also changes the following:

- Factor 1 and factor 2 (Factor 1 and factor 2 are used in compare operations. The GSORT program compares the data in the field specified as factor 1 with the data noted as factor 2.)
- Normal and opposite key fields
- Input field characters that condition forced key fields

*Note: With an alternate collating sequence on the entire key field, do not use packed or unpacked decimal fields as factor 1 and factor 2 in an Include or Omit Record Selection specification (P or U should not appear in column 8).*

See “Coding Record Selection Information” for information on factor 1 and factor 2. See “Coding Field Selection Information” for details about types of key and data fields.

## Using Alternate Collating Sequences for Specified Key Fields

When you use an alternate collating sequence on specified key fields, the ALTSEQ statement changes only specified normal and opposite key fields.

For normal and opposite key fields, enter an F in column 26 of the Header specification. Then enter your ALTSEQ statement. Enter an A in column 20 of the key field statement of the Field Selection specification.

Use the following guidelines when planning to use an alternate collating sequence on specified key fields:

- Record selection (including and omitting records) and conditional force (replacing one or all characters) are based on an input record that has not been changed by the alternate collating sequence.
- A key field that has an A specified in column 20 should not be packed or unpacked (P or U should not appear in column 8).
- You can specify that factor 1 and factor 2 be packed or unpacked in the Include and Omit Record Selection specifications by entering P or U in column 8.
- When you specify an alternate collating sequence for a particular field, that field is changed according to the alternate collating sequence whenever it is used again as a key field for that record type. This repetition occurs only if the same input field is specified more than once as a key field.

See “Coding Record Selection Information” for information on omit and include sets as well as factor 1 and factor 2. See “Coding Field Selection Information” for details about types of key and data fields.

## Using the ALTSEQ Statement to Make Characters Equal

When you move a character into the sequence position normally assigned to another character, both the new and the original character occupy the same position and are considered equal. If you do not want the two characters to be equal, move the character that normally occupies that position. However, if you want one character to be considered the same as another character, both characters must hold the same position in the collating sequence.

For example, you might want a blank to be considered a zero. You need to define an alternate collating sequence in which the blank is the same as the zero because it holds the same position in the sequence. The ALTSEQ statement would be

```

          1 1 1 1 1 1 1
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
A L T S E Q   4 0 F 0

```

In this situation, whenever a blank is read and used in a comparison, it is considered a zero. Thus, if you were comparing numbers to 0017 to find an equal condition, 0017 and bb17 (where b = blank) both compare as equal to 0017.

*Note: Be careful when you use packed or unpacked fields to make characters equal with an alternate collating sequence. A portion of the packed or unpacked characters could be translated and become invalid numbers.*

## Coding Record Selection Information

The Record Selection specification is used to determine the records that are included in the GSORT job. All Record Selection specifications must precede any Field Selection specification.

*Note: No Record Selection specification is needed if both of the following conditions are true:*

- *All the records in a file are to be sorted.*
- *All the records are the same type.*

The following is an example of the fixed format of the Record Selection specification. Table E-3 summarizes the column entries. The entries are described more fully in the text that follows the table.

Sequence Number	Include/Omit Specification	Factor 1		Relationship (I, O, A, OR)	Factor 2 Location	Comments	
		Field Location	Start			End	Factor 2 Location
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							
73							
74							
75							
76							
77							
78							
79							
80							

Table E-3. Record Selection Specification Entry Table

Columns	Valid Entries	Purpose of Entry
1-5	00001-99999	Sequence number of the specification.
6	I	Indicates this record is to be included in the sort.
	O	Indicates this record is to be omitted from the sort.
7	Blank	Indicates this is the first specification in a set of Include or Omit Record Selection specifications.
	A	Indicates an AND relationship with the previous include or omit option.
	O	Indicates an OR relationship with the previous include or omit option.

continued

Table E-3. Record Selection Specification Entry Table (cont.)

Columns	Valid Entries	Purpose of Entry
	*	Comments.
8	C	Designates comparison of both zone and digit portions of each character.
	Z	Designates comparison of only the zone portion of each character.
	D	Designates comparison of only the digit portion of each character.
	P	Designates that data are in packed decimal format and are signed.
	U	Designates that data are in unpacked decimal format and are signed.
9-12	1-4096	Indicates the starting position of the factor 1 field in the input record. These columns can be blank if the factor 1 field is only one position long.
13-16	1-4096	Indicates the ending position of the factor 1 field in the input record.
17-18	EQ	Factor 1 is equal to factor 2.
	NE	Factor 1 does not equal factor 2.
	LT	Factor 1 is less than factor 2.
	GT	Factor 1 is greater than factor 2.
	LE	Factor 1 is less than or equal to factor 2.
	GE	Factor 1 is 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.
	K	Factor 2 is a keyword.
20-23	1-4096	Designates the starting position of the factor 2 field in the input record. These columns can be blank if the factor 1 field is only one position long.
24-27	1-4096	Designates the ending position of the factor 2 field in the input record.
20-39	Any characters	Identifies the characters that make up the factor 2 constant.
40-72	Any characters	Comments.



### Sequence Number (Columns 1 through 5)

Sequence numbers are optional. If entered, they must be in ascending order. If the numbers are not in ascending order, when the specifications are being processed an error message is placed next to the statement that is out of sequence.

Any item in these columns that is not a number is ignored by the program.

### Include/Omit Specifications (Column 6)

This column identifies the include or omit action to be taken by the GSORT job.

To specify an include option, enter an I in column 6. To specify an omit option, enter the letter O. When sorting a file containing only one type of record, if you want to include all records, leave columns 7 through 39 blank.

The include option identifies which records in a file should be sorted. You can specify either an include-all or a conditional include option.

The omit option indicates which records are not to be sorted. The option is useful when you have many types of records to sort and only a few to exclude.

### Using the Include-All Option

An include-all option usually follows an omit option to indicate to the GSORT program that all records not described by any preceding omit option are to be included and sorted. An include-all is usually used when all the records in a file have the same record type. Such records must have the same Field Selection specifications.

Each GSORT job can use only one include-all option. If it is used, it must be the last Record Selection option for that job.

### Using the Conditional Include Option

Use the conditional include option to cause the GSORT program to test data in the input records to see if they meet a particular condition before they are included. In the Record Selection specification, you must describe the fields in the record to be tested.

### Mixing the Include and Omit Options

When you mix include and omit options you should be particularly careful about their order. The GSORT program processes the options in the order in which they are coded. You must define the omit option before you define the include option for a record type. The include-all option tells the GSORT program to sort all the records that are not specified by the omit option.

Often the records in a record type have at least one characteristic in common. To sort all but a few records in a file, use an omit option followed by an include option and other associated Record Selection specification options for each type of record you want to sort.

The following are four guidelines for using both the include and omit options:

- A Field Selection specification must end all sets of include options.
- The last option specified must be an include option.
- Every group of omit options, also known as an omit set, must be followed by an include option.
- Omit options are never followed directly by Field Selection specifications.

See “Sorting a File with Conditional Include and Omit” for examples of programs that use conditional include, conditional omit, and a mixture of include and omit options.

### Continuation or Comments (Column 7)

An include or omit set consists of one or more records where each record specifies one include or omit option. The entry in column 7 indicates the relationship of a record to other include or omit records within the set.

Entry	Meaning
Blank	Indicates the first of a series of include or omit entries.
A	Indicates an AND relationship with the previous include or omit option.
O	Indicates an OR relationship with the previous include or omit option.
*	Indicates a comment line, and the line is to be ignored by the program. Comments are printed only if column 27 of the Header specification contains a zero or a blank.

For the continuation entries, noted by the letter A or O in column 7, the entry in column 6 must match all other records in the set.

Table E-4 shows the valid combinations in columns 6 and 7 for the include option. Column 6 identifies the set type: include. The relationship of the record to other records in the set (first in a series, a Boolean AND or OR) is given in column 7.

Table E-4. Coding for Include Sets

Relation	Entry in Column 6	Entry in Column 7	Explanation
AND	H, F, or O		Header specification, Field specification, or Omit Record Selection specification.
	I	Blank	Indicates a new record type.
	I	A	Indicates that this specification describes the same record type as the previous specification.
	F		Field specification or specifications.

continued

Table E-4. Coding for Include Sets (cont.)

Relation	Entry in Column 6	Entry in Column 7	Explanation
OR	H, F, or O		Header specification, Field specification, or Omit Record Selection specification.
	I	Blank	Indicates a new record type.
	I	O	Indicates that this specification describes a different record type than the previous specification.
	F		Field specification or specifications.
AND and OR	H, F, or O		Header specification, Field specification, or Omit Record Selection specification.
	I	Blank	Indicates a new record type.
	I	O	Indicates that this specification is different from the record type of the previous specification, but has the same Field specifications as the previous specification.
	I	A	Indicates that this specification is the same as the previous specification or specifications and has the same Field specifications as the previous specifications. A record type with different Field specifications must be defined in a new include set.
	F		Field specification or specifications for the record types.
Including only one record type (implies include-all records)	H		Header specification.
	F		No Record Selection specifications are included. Field specification or specifications.

continued

Table E-4. Coding for Include Sets (cont.)

Relation	Entry in Column 6	Entry in Column 7	Explanation
Include-all records	H, F, or O		Header specification, Field specification, or Omit Record Selection specification.
	I	Blank	Indicates all records that have not been described in any of the preceding Record Selection specifications should be sorted. Records identified in this manner must have identical Field specifications.
	F		Field specification or specifications.

Records not described in include sets are not sorted. All include sets must end with one or more Field specifications. An include set can be followed by another include set or an omit set.

**Example: Include Set with an OR Relationship**

Input records must meet one of two criteria to be included in the output record: they must have either a status of S or a quantity of less than 200. The Field specifications note the sort field and the fields in the output record.

```

      1      2      3      4      5      6
123456789012345678901234567890123456789012345678901234567890
$GSORT
HSORTR  20A      0X 40
I C 60 60EQCS      INCLUDE 'S'HIPPED RECORDS
IOU 50 59LTC      200      OR WHERE QTY <200
FNU 40 49      SORT ON ITM NBR 40..49
FDC 60 60      1ST DATA FLD: STATUS FLAG
FDU 40 49      2ND DATA FLD: ITM NBR
FDC 10 29      3RD DATA FLD: LAST NAME
    
```

**Example: Include Set with an AND and OR Relationship**

In this example, there are two include sets. The first set states that records with a status of S and a quantity less than 200 or greater than 500 should appear in the output file. Each include option applies to the same record type. For the second set, all records with a status of W also appear in the output file. For each set, the Field specifications note the sort field and the fields in the output record.

## Using GSORT to Code and Run Sort Programs

```

1          2          3          4          5          6
123456789012345678901234567890123456789012345678901234567890
$GSORT
HSORTR    20A          0X 40
I C 60 60EQCS          INCLUDE 'S'HIPPED RECORDS
IAU 50 59LTC          200    WHERE QTY <200
IOU 50 59GTC          500    WHERE QTY >500
FNU 40 49          SORT ON ITM NBR 40..49
FDC 60 60          1ST DATA FLD: STATUS FLAG
FDU 40 49          2ND DATA FLD: ITM NBR
FDC 10 29          3RD DATA FLD: LAST NAME
I C 60 60EQCW          INCLUDE 'W'AIRING RECORDS
FDN 10 29          SORT ON LAST NAME
FDC 10 29          1ST DATA FLD: LAST NAME
FDC 60 60          2ND DATA FLD: STATUS FLG

```

See "Sorting a File with Conditional Include and Omit" for other examples of using include sets.

Table E-5 shows the valid combinations in columns 6 and 7 for the omit option. Column 6 identifies the set type: omit. The relationship of the record to other records in the set (first in a series, a Boolean AND or OR) is given in column 7.

**Table E-5. Coding for Omit Sets**

Relation	Entry in Column 6	Entry in Column 7	Explanation
AND	H or F		Header specification or Field specification.
	O	Blank	Indicates a new record type.
	O	A	Indicates that this specification describes the same record type as the previous specification.
OR	H or F		Header specification or Field specification.
	O	Blank	Indicates a new record type.
	O	O	Indicates that this specification describes a different record type than the previous specification.
AND and OR	H or F		Header specification or Field specification.
	O	Blank	Indicates a new record type.

continued

Table E-5. Coding for Omit Sets (cont.)

Relation	Entry in Column 6	Entry in Column 7	Explanation
	O	A	Indicates that this specification describes the same record type as the previous specification.
	O	O	Indicates that this specification describes a different record type than the previous specification.

There are no Field specifications in omit sets. Each omit set must be followed by one or more include sets or an include-all set.

See "Sorting a File with Conditional Include and Omit" for examples of using omit sets.

### Interpretation of Data (Column 8)

The entry in this column specifies whether the conditions for inclusion or deletion compare alphanumeric characters or numeric data. The entry also determines if a GSORT program compares zones, digits, or both.

GSORT sees characters as EBCDIC characters composed of 8 bits that make up a byte of data. Each 8-bit character has two parts: the high-order 4 bits are the zone portion and the low-order 4 bits are the digit portion. For example, the character B appears to the system as follows:

Zone	Digit
1100	0010

For alphanumeric data, choose from the following options:

Entry	Meaning	Maximum Field Length
C	Use both the zone and digit portions of the characters.	256 characters
Z	Use only the zone portion of the character. All characters with identical zone portions look alike and compare as equal.	1 character

Entry	Meaning	Maximum Field Length
D	Use only the digit portions of the characters. All characters with identical digit portions look alike and compare as equal.	16 characters

For numeric data, choose from the following options:

Entry	Meaning	Maximum Field Length
P (packed data)	Use both the zone and digit portions of the number, with each zone and digit representing a value from 0 through 9.	8 bytes or 15 digits and a sign
U (zone data)	Use the digit portions of numbers only, with a sign in the last digit.	15 digits and a sign

The letter P or the letter U must not be specified if an alternate collating sequence is specified in the Header specification.

To select records based on a binary value, you can use one or both of the following methods of inclusion.

For binary values whose hexadecimal equivalents can be represented by one or two characters in the 64-character set, use an include option (I in column 6) and either of the following two methods:

- Place a C in column 8 of the Record Selection specification to indicate that both the zone and digit portions of a character should be compared.
- Place a Z in column 8 of the Record Selection specifications to indicate that only the zone portion should be compared, and use an AND option (place IAD in columns 6 through 8 of the Record Selection specification) to indicate that the digit portion should then be compared.

For binary values whose hexadecimal zone portions do not appear in the 64-character set, use two include options for the zone portion, using an AND option with an include digit for the digit portion.

### Factor 1 Location (Columns 9 through 16)

Factor 1 fields are used in compare operations. Factor 1 indicates the location of fields in the input records that the GSORT program tests to see if a condition exists. Factor 2 specifies the data against which these fields are tested. The GSORT program compares the value in the factor 1 field with the data in factor 2.

You specify in these columns the locations of the factor 1 fields in the input records.

In columns 9 through 12, identify where the factor 1 field starts in the record.

In columns 13 through 16, identify where the factor 1 field ends.

If you have more than one factor 1 for the records you are describing, identify the location of each and do the following:

- Use a separate line in the Record Selection specification to describe each test the GSORT program will do.
- Put the letter O (for OR) in column 7 of each statement that defines a different record type from that defined in any previous statement.
- Put the letter A (for AND) in column 7 of every specification except the first, to tell the GSORT program that all the specifications apply to the same record type.

You must right-justify the factor 1 field entries: End the Start-location entry in column 12; end the End-location entry in column 16.

If factor 1 is only 1-character long, you need identify only the end position. Leave columns 9 through 12 (Start) blank and enter the number of the record position that contains the character in columns 13 through 16 (End).

The length of the field must match the length of the second operand in the comparison. Numeric fields cannot have more than 23 digits.

Factor 1 can contain from 1 to 256 characters. However, factor 1 cannot be longer than the length of the records being sorted or reformatted. The column 8 entry controls the length of factor 1, as follows:

Column 8	Maximum Factor 1 Field Length
C	256 characters
Z	1 character
D	16 characters
P	8 characters
U	16 characters

The length of factor 1 fields is also controlled by factor 2 in the following situations:

- When factor 2 is a constant, the length of factor 1 must be 20 characters or fewer.
- When factor 2 is a keyword, the length of factor 1 must be 6 if the keyword is UPDATE, and 2 if the keyword is UMONTH, UDAY, or UYEAR.

### Relation (Columns 17 and 18)

These columns specify the criteria for the relationship between the value in factor 1 fields and the data in factor 2.

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.



Entry	Meaning
GE	Factor 1 must be greater than or equal to factor 2.

The entries EQ and NE are the only entries you can use to compare zone portions of characters (Z in column 8).

If you specify an alternate collating sequence and place an S in column 26 to indicate that the alternate collating sequence is to be applied on the entire key field, both factor 1 and factor 2 are changed to the alternate collating sequence before the comparison is made.

### Factor 2 Compare Data (Column 19)

The entry in column 19 specifies whether the data in factor 2 is a literal or a constant, a field in the input record, or a date keyword.

Entry	Meaning	Related Entries
C	Factor 2 is either a literal or a constant.	Enter the constant in columns 20 through 39, starting in column 20.
F	Factor 2 is a field in the input record.	Use columns 20 through 27 to identify the location of the factor 2 field in the records. As desired, enter the name of the factor 2 field in columns 28 through 39.
K	Factor 2 is a keyword.	Start the keyword in column 20. Enter a C in column 8 so that both the zone and digit portions are compared.

### Factor 2 Location (Columns 20 through 27)

This entry identifies the location of factor 2 when it is another field in the input records.

Use columns 20 through 23 to identify the starting position (Start-location) of the field. Use columns 24 through 27 to identify where the field ends (End-location). Both the End-location and Start-location entries must be right-justified.

Factor 2 can be used to compare a field containing 1-character. In that case, enter the End-location (columns 24 through 27) only; leave the Start-location blank.

### Compare Field Name (Columns 28 through 39)

If factor 2 is a field in the input record, columns 28 through 39 can contain the name of the field. This entry is optional and treated as documentation.

## Compare Literal or Constant (Columns 20 through 39)

If factor 2 is a literal, enter the literal in columns 20 through 39 using the following guidelines:

- If the literal is alphanumeric, the data must be left-justified.
- If the literal is numeric, it can be entered anywhere in columns 20 through 39.
- The literal must be the same length as the value with which it is compared.
- When you use signed packed (numeric) literals, the leftmost or rightmost character (but not both) can contain its sign (+ or -).

When you use signed ALPHA literals, the sign can be the leftmost character. For negative signed ALPHA literals, the rightmost character (number) appears as shown in Table E-6. (The zone portion contains the negative sign of the entire number. The digit portion contains the numeric value of the last digit of the number.)

Table E-6. Coding for Negative Unpacked Numbers

Number	Replace by	Hexadecimal Equivalent	Zone Portion	Digit Portion
0	(blank)	A0	0110	0000
1	J	D1	1101	0001
2	K	D2	1101	0010
3	L	D3	1101	0011
4	M	D4	1101	0100
5	N	D5	1101	0101
6	O	D6	1101	0110
7	P	D7	1101	0111
8	Q	D8	1101	1000
9	R	D9	1101	1001

If factor 2 is a constant, it can be unsigned (character, zone, and digit data types) or signed (unpacked and packed data types). You can use any arrangement of characters and blanks as entries. Enter the constant in columns 20 through 39, starting in column 20. Determine the length of the constant as follows:

- The constant must be the same length as the factor 1 field. If the constant is longer than the factor 1 field, GSORT prints a warning message.
- If factor 1 contains a packed decimal number, the length of the constant, including the sign, must be twice the length of the factor 1 field.
- If the constant is an alphanumeric constant, it must be the same length as factor 1 and must start in column 20. For alphanumeric constants, a D in column 8 indicates that only the digit portion of a character is to be used in the compare operations.

- Numeric constants (U or P in column 8) must be right-justified within the field length specified in factor 1, or within twice the field length if factor 1 is a packed decimal number.

If factor 1 is a packed decimal number, the last character in the constant must be its sign, plus (+) or minus (-).

If factor 1 is an unpacked decimal number and the constant is a negative number, the last character in the constant must indicate both the numeric value of the last digit and the negative sign for the entire constant. (See Table E-6.)

### Compare Date Keyword (Columns 20 through 39)

When factor 2 is a date keyword, the date keyword starts in column 20; all unused columns through column 39 should be left blank. The factor 2 keyword can be a maximum of 6 characters long.

The factor 2 keyword can be one of the following:

Keyword	Part of Program Date	Factor 1 Field Length
UPDATE	Entire program date	6 characters
UMONTH	Month portion of program date	2 characters
UDAY	Day portion of program date	2 characters
UYEAR	Year portion of program date	2 characters

If the UPDATE keyword is used, the program date must be in the same format as that of the date contained in the input records. Use the international date format (yymmdd).

If the program date and the input records date are not in the international date format, you must use the keywords UYEAR, UMONTH, and UDAY to compare the three elements separately.

### Comments (Column 40 through 72)

These columns can be used to enter optional comments and to document the program. If you specify that the GSORT job print your specifications (the column 27 entry of the Header specification is zero or blank), the comments you include in these columns are printed.

## Coding Field Selection Information

The Field Selection specification is used to define the key and data fields. These specifications identify how to arrange and format input records in the work and output files.

For all types of sorted output, Field Selection specifications describe the fields the GSORT program uses to sort the records (the key fields). For regular sorts (SORTR) and summary sorts (SORTRS), Field Selection specifications describe the data that



Table E-7. Field Selection Specification Entry Table (cont.)

Columns	Valid Entries	Purpose of Entry
9-12	1-4096	Indicates the starting position of the field in the input record. These columns can be blank if the factor 1 field is only one position long.
13-16	1-4096	Indicates the ending position of the field in the input record.
17	Any character	Identifies either the character that is to be changed during the GSORT, or the overflow indicator when a summary field is used.
18	Any character	Identifies either the forced character that replaces the character identified in column 17, or the initialized value of the overflow field.
19	Blank	If the preceding specification has an F in column 7, indicates that this is not a continuation of that specification.
	Any character other than blank	If the preceding specification has an F in column 7, indicates that this is a continuation of that specification.
20-22	1-256	Indicates the field length of the overflow field in a summary GSORT.
	A (column 20 only)	Indicates the use of an alternate collating sequence.
23-39	Blank	Not used.
40-72	Any characters	Comments.

### Sequence Number (Columns 1 through 5)

Sequence numbers are optional. If entered, they must be in ascending order. If the numbers are not in ascending order, when the specifications are being processed an error message is placed next to the statement that is out of sequence.

Any character in these columns that is not a number is ignored by the program.

### Field Selection Specifications (Column 6)

An F must appear in column 6 to identify a Field Selection specification.

## Field Type or Comments (Column 7)

The entry in this column identifies the field type and subtype.

Entry	Meaning
N	Indicates a normal key field. Sort this field so that the data from the field are in the order specified in column 18 of the Header specification.
O	Indicates an opposite key field. Sort this field so that the data from the field is in the order opposite that specified in column 18 of the Header specification.
F	Indicates a forced key field. Before sorting, change the key field according to the entries in columns 9 through 19. Use one of the following forced key fields:  Forced without condition: Forces a character into a key field before the records are sorted.  Forced with conditions: Forces a character into a key field only when a condition is met.  Force-all. Forces a character into a key field before the records are sorted if the key field does not contain one of several entries.
D	Indicates a data field. Use this entry for SORTR and SORTRS jobs only. If you use a D entry in a SORTA job, the specification is treated as a comment.
S	Indicates a field containing data that is to be totaled for all records with identical key fields. Use this entry for SORTRS sorts only. If you use an S entry in a SORTA sort, the statement is treated as a comment. If you specify an S in a SORTR sort, the field is treated as a normal data field.
*	Indicates a comment line; the comment is ignored by the program.

## Defining Types of Key Fields

You can define the following types of key fields:

- Normal key fields
- Opposite key fields
- Forced key fields

Any field, containing 1 to 256 characters, in the input record can be used as a key field to control the sorting of the records. You can use one or several key fields in a GSORT job. When you sort the records using more than one key field, the major key field is always the first key field you define for a record type. It is used first in the sorting process. Minor key fields are the other key fields you define for a record type. You determine the hierarchy of the key fields by the order in which you define them.

To include normal key fields and opposite key fields in the output file, you must describe the field twice: once as a key field and once as a data field.

## Using GSORT to Code and Run Sort Programs

---

When the input file has more than one type of record, you can do one of the following:

- You can make the key fields the same or different for each type of record. However, be sure you specify the correct data type, such as numeric or character data, in the Field Selection specifications.
- You can make the length of the key field the same or different for each type of record. (See the explanation for the Key Field Lengths columns in the Header specification for more information.)

### Normal Key Fields

A normal key field (N) is any field in the input records that the GSORT program uses to sort the records. The data in that particular field are sorted in ascending or descending order, whichever is specified in the Header record.

### Opposite Key Fields

Use opposite key fields (O) to define a field sequence that is the opposite of the sequence specified for other fields in the Header record. For example, the field sequence will be in ascending order if you specified descending order in the Record-Sorting Order column of the Header specification.

### Forced Key Fields

Use forced key fields (F) to force a record to take a particular position when it is sorted. For example, if you were sorting by three criteria (last name, first name, and middle name), you could use a forced key field to determine how to sort records without middle names.

Each forced key field can be only 1-character long, but you can specify more than one forced key field for a given field. To do so, you define a Field Selection specification for each single character constant you want forced into a key field.

You can specify three types of forced key fields:

- Unconditional
- Conditional
- Force-all

### Unconditional Forced Field

Use an unconditional forced field to automatically place a specified character constant into the first position of the GSORT key field work record.

### Conditional Forced Field

Use a conditional forced field to test a character in the input records and then to force a character constant into the GSORT key field only if the test is successful. You can

make the test on any character in the input record. You can compare the entire input character or just the zone or digit portion.

As you define a conditional forced field, you can specify that the GSORT program force the constant either into the previously defined key field position or into the next available key field position.

### Force-All Field

Use a force-all field after a series of Field Selection specifications that contain conditional forced fields. This feature allows you to specify a character constant to be forced into the key field when you already have a forced condition but when none of the preceding conditional force test conditions are met.

At least one conditional force specification must come before the force-all specification.

### Specifying Types of Data Fields

Data fields are not involved in the sorting process and are not changed by the GSORT program. You can specify one of three types of data fields:

- Normal data fields
- Summary data fields (fields containing data to be totaled)
- Forced data fields

### Normal Data Fields

Normal data fields apply to regular (SORTR) and summary (SORTRS) GSORT jobs only. These fields are included in the sorted records, but are not used in the sorting process. Data are written into the output records in the order in which you define the fields on the Field Selection specification. Always specify data fields after key fields.

When you give a file more than one type of record, you can

- Have different numbers of data fields.
- Have different total lengths of all data fields for all record types. GSORT places blanks to the right of shorter records so that all records are the same length.

If you specify data fields for a SORTA job, these specification statements are ignored.

### Summary Data Fields

Summary data fields are used to hold accumulated totals. These fields can be specified only in summary (SORTRS) sort jobs.

If you have multiple input record types, you must ensure that the summary output data for each record type is defined to match any summary data fields defined in the other record types.



## Using GSORT to Code and Run Sort Programs

---

You define summary data fields like normal output data fields:

- Column 6 must be an F to indicate a Field Selection specification.
- Column 7 must be an S to indicate a summary field.
- Column 8 defines the data type of the field to be added.

You can accumulate totals for a maximum of 24 different output fields in SORTRS job.

The data types (selected in column 8) used for normal data fields are also used for summary data fields. The data type entry you use determines the method of adding the data.

The Field Location field, columns 9 through 16, must indicate the start and end positions of the summary data field in the input record. As with normal data fields, you do not need to specify the start position for a field that is only one position long.

### Forced Data Fields

When you run a regular sort (SORTR) or a summary sort (SORTRS), you can force a character constant at selected locations in the output data.

### Data Type (Column 8)

The entry in this column indicates what portion of the input record characters you want the GSORT program to use when sorting records.

When you compare letters, numbers, and special characters, you must designate either unsigned or signed data in column 8.

Unsigned data is made up of alphanumeric or alphabetic data and includes special characters such as mathematical symbols and punctuation marks. For unsigned alphanumeric data, choose from the following options:

Entry	Meaning	Maximum Field Length
C	Use both the zone and digit portions of the characters.	256 characters
Z	Use only the zone portion of the character.	1 character
D	Use only the digit portions of the characters.	16 characters

Signed data consist of either positive or negative numbers. Signed numbers are ordered by both their numeric values and their signs, either plus (+) or minus (-). Signed numbers are of two types: unpacked or packed.

For signed numeric data, choose from the following options:

## Using GSORT to Code and Run Sort Programs

Entry	Meaning	Maximum Field Length
P (packed data)	Use both the zone and digit portions of the number, with each zone and digit representing a value from 0 through 9.	8 characters (15 digits and a sign)
U (zone data)	Use the digit portions of numbers only, with a sign in the last digit.	16 characters

Unpacked decimal numbers are made up of 8 bits. Each character in an unpacked data field has its numeric value in the digit portion of the character. The sign of the number is in the zone portion of the rightmost character in the field.

Each digit of a packed decimal number is represented by 4 bits. A numeric value is placed in both the zone and digit portions of each character in a field, except the rightmost character. The sign of the number is placed in the rightmost digit of the field.

For forced data, the following option applies:

Entry	Meaning	Maximum Field Length
V	Force a data character constant into the data field.	1 character

Table E-8 shows the combinations of entries you can make in columns 7 and 8.

**Table E-8. Possible Entry Combinations (Columns 7 and 8)**

Column 7	Column 8	Maximum Field Length
N or O	C	256
	Z	1
	D	16
	P	8
	U	16
F	C	1
	Z	1
	D	1
D	C	256
	Z	1
	D	16
	P	8
	U	16
	V	1

continued

Table E-8. Possible Entry Combinations (Columns 7 and 8) (cont.)

Column 7	Column 8	Maximum Field Length
S	C	256
	Z	1
	D	16
	P	8
	U	16
	V	1
*		Comments

*Note: Packed or zone data cannot be used if an alternate collating sequence is specified in the Header specification.*

For a summary data field, the data type entry determines the method of adding the data:

- For P (packed) data type, both the zone and digit portions of the input data are added, along with an optional sign.
- For U (unpacked) data type, only the digit portions of the input data are added, along with an optional sign.
- For D (digit) data type, only the digit portions of the input data are added, without a sign involved.
- For C (character) data, the entire character is added, without a sign involved.
- For Z (zone) data type, the zone portion of the input data is added, without a sign involved.

## Field Location (Columns 9 through 16)

The entries in these columns give the Start- and End-locations of the fields in the input record to be included in the output record. The fields can be either key fields and data fields. The order in which you list the field locations determines their order in the sorted output records.

In columns 9 through 12, identify where the field starts. In columns 13 through 16, identify where the field ends. Entries for the Start- and End-locations should be right-justified.

If you describe fields that are only 1 character long, leave columns 9 through 12 blank; enter only the End-location of the character in columns 13 through 16. Right-justify the entry.

The length of the field depends on the entry in column 8. For the maximum field lengths allowed for each entry, see the column 8 description earlier in this section.

### Compare Character (Column 17)

Use an entry in this column to conditionally force a character into a key field or to define a character as a summary overflow indicator.

#### For a Forced Key Field

If column 7 contains the letter F, the forcing should be done only if the value of the 1-character-long field, whose End-location is noted in columns 13 through 16, is equal to the character in column 17.

The GSORT program checks to see if the field in the input record contains the character given in column 17. If it does, the character in column 18 replaces the specified character in the key field.

A series of 1-character-long fields can be compared and replaced by the entry of a continuation character in column 19.

#### For a Summary Overflow Indicator Field

If column 7 does not contain the letter F, a nonblank value in column 17 specifies a summary overflow character. Enter the character you want to place in the output record if overflow occurs.

A blank in this column indicates that the summary overflow indicator is an asterisk (\*).

As an additional feature, you can use column 18 to specify the character to use if an overflow does not occur. (See "Detecting Summary Data Overflow".) Summary overflow indicator fields are valid only for summary (SORTRS) jobs.

### Forced Character (Column 18)

This column contains the character to be forced. An entry should be made in this column only when the letter F is in column 7, or the letter D in column 7 is accompanied by the letter V in column 8.

The character in column 18 either replaces the key field character specified in column 17, adds a new character to the key field, or adds a new character to the data field.

You can force a character constant into the output to indicate when an overflow occurs and when it does not occur. The character in column 17 is used when overflow occurs. The character in column 18 is used when the overflow does not occur. (See "Detecting Summary Data Overflow".)

Any character can be used. Summary overflow indicator fields are valid only for summary (SORTRS) jobs.

### Continued Forced Character in Forced Key Field (Column 19)

You can only force (replace) one character at a time. However, you can replace a series of characters in the same key field by using continuation records. Each character must have its own continuation record that defines the character and the character to replace it. To create a continuation record, enter any character in column 19.

The first forced character in the key field always has an F in column 7, its End-location in columns 13 through 16, and entries in columns 17 and 18. Any subsequent forced character has an F in column 7, its End-location in columns 13 through 16, and entries in columns 17, 18, and 19.

### Overflow Field Length (Columns 20 through 22)

These columns contain the length of the overflow field. Use these columns with a GSORT job that accumulates totals (SORTRS) to prevent the occurrence of an overflow condition in a summary data field. You can increase the length of the field by placing the entry for the new length in these columns.

In these columns, define the amount of memory the GSORT program should reserve for any summary data totals. The length of this field cannot be longer than the maximum length allowed for the input data type specified in column 8, as shown in Table E-9.

**Table E-9. Maximum Field Lengths for Input Data Types**

Data Types	Maximum Field Length
C (character)	256 characters
D (digit)	16 characters
Z (zone)	1 character
U (unpacked)	16 characters
P (packed)	8 characters

The overflow field length entry should

- Equal the length of the summary data field plus the expected overflow length
- Be right-justified to column 22
- Not exceed the maximum field length

For each record type in a GSORT job, a maximum of 24 fields can be totaled.

If packed decimal fields are summarized, columns 20 through 22 should specify the number of bytes of packed data contained in the field.

### Preventing Summary Data Overflow

If the length of the field that will hold the total is not big enough, any higher order digits are lost, with no indication of the loss. To prevent this situation from occurring, designate the largest output area that could be needed for a particular total.

### Detecting Summary Data Overflow

With GSORT, you can force a character constant into the output to indicate when overflow occurs. Subsequent programs can then read the GSORT output file and test for the presence of this character constant to see if overflow occurred for the summary field.

You can specify only one summary overflow indicator per record type. If you specify more than one per record type, only the first one has any effect on the output record.

To specify an overflow indicator field, do the following:

1. Fill in columns 1 through 6 on the Field Selection specification.
2. Enter an S (for summary) in column 7 of the Field Selection specification.
3. Enter a V (for forced field) in column 8 of the Field Selection specification.
4. Leave columns 9 through 16 of the Field Selection specification blank.
5. Enter the character to which the overflow indicator field will be set (column 17) if overflow occurs in any summary data field.

If overflow occurs and no overflow character is identified (that is, column 17 contains a blank) the system sets the overflow indicator field to an asterisk (\*).

6. Enter a character in column 18 to which the overflow indicator field is set if an overflow does not occur in any summary data fields.

If none of the summary data fields overflows, the overflow indicator field contains the character specified in column 18. If column 18 contains a blank, the overflow indicator field is set to a blank.

### Alternate Collating Sequences (Column 20)

When column 26 of the Header specification contains an F, column 20 must contain an A for any normal or opposite key field that is to be sorted by the alternate collating sequence.

If you specify an alternate collating sequence for a particular field, that field is changed according to the alternate collating sequence whenever the field is used again as a key field for that record type.

### Columns 23 through 39

These columns are not used.

### Comments (Columns 40 through 72)

These columns can be used to enter optional comments and to document the program. If you specify that the GSORT job print your specifications (the column 27 entry of the Header specification is zero or blank), the comments you include in these columns are printed.

### Examples of GSORT Programs

The following examples demonstrate how to code using the GSORT compiler control record syntax.

In the examples, numeric data is coded as unpacked data. All other data is coded as character data. This means that fields such as customer number or item number are unpacked data and fields such as customer name or status are character data.

Unpacked data is output with leading zeroes. If a customer number is 222 and the field is 9 characters long, the output is "000000222". For readability, leading zeroes are replaced by blanks in all input and output examples.

### Sorting without Record Selection Specifications

1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
10000	\$GSORT					
12000	H SORTR	6A	X 128			SORT PAYROLL ASCENDING
14000	FNU	2	7			CONTROL FIELD IS EMPLOYEE ID
16000	FDC	1	128			PUT ALL DATA INTO OUTPUT FILE

#### Header Specification

In the Header specification, the job is defined as a regular sort (SORTR). The length of the key field is 6. The items are to be sorted in ascending order. Although the key field is dropped in column 28, its contents are specified as a data field in the Field Selection specification. Columns 30 through 32 show the number of characters in each record of the output file. The comments, starting in column 44, clarify what the sort will do.

#### Record Selection Specification

Because all records from the input file are included in the sort, there are no Record Selection specifications. This situation is referred to as an implied include-all.

#### Field Selection Specifications

There are two Field Selection specifications. The first identifies the key field for the sort. The second indicates which fields should be included in the output records.

The first Field Selection specification identifies a normal key field whose data the GSORT program should interpret as character data (both the zone and digit portions are

compared). The entries in column 12 and 16 identify the location of the key field. (The GSORT program compares the zone and digit portions of each character in positions 2 through 7 to determine the order of the records in the output file.) A comment begins in column 40.

In the second Field Selection specification, the D in column 7 specifies that the fields in positions 1 through 128 are data fields in the output file. The GSORT program should interpret the data as character data. The entries in columns 9 through 16 indicate the location of the fields in the input record that are considered data fields in the output file. In this case, all the data in the input records are written into the output file. A comment begins in column 40.

### Sorting a File with Selected Records

```

      1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012
$GSORT
HSORTR      3A          X 10  N          SORT CHARGEREC INFO
I U 16 22GEC0003000          INCLUDE AMOUNTS OF $30.00 OR MORE
FNU 3 5          CONTROL FIELD IS CHARGE CODE
FDU 3 5          FIRST DATA FIELD IS CHARGE CODE
FDU 16 22          SECOND DATA FIELD IS AMOUNT
    
```

In this example the records in the input file have the following format:

```

|Record Code|Charge Code|Description|Amount|Guest Name|
1         2 3         5 6         15 16 22 23         40
    
```

For this GSORT job, you want to select only items that are charged for an amount of \$30 or more. You want to sort the items in ascending order, with the charge code as the key field. For each item over \$30, you want only the information in the charge code and amount columns included in the output file.

#### Header Specification

The Header specification identifies this as a regular sort (SORTR), in ascending order. The 3 in column 17 indicates the number of characters in the field in the input records that are to be used to sort the records (the Charge Code field). Information in the key field is to be dropped from the output file. The output record will contain 10 characters. An error message will not be displayed if there are no input records that meet the conditions you specify in the Record Selection Statements. The comment begins at column 44.

#### Record Selection Specification

The Record Selection specification indicates that the GSORT program should include only those records whose contents in positions 16 through 22 of the input records are equal to or more than \$30. The letters GE in columns 17 and 18 specify that the amounts to be selected should be greater than or equal to the amount given beginning at column 20.



Column 8 specifies that data in positions 16 through 22 of the input records are unpacked data. Columns 11 and 12, and 15 and 16 respectively, show the starting and ending positions of the field to be examined in the input record. The letter C in column 19 identifies Factor 2 as a constant. The optional comment begins at column 40.

### Field Selection Specifications

You need three Field Selection specifications for this job. The first identifies the field as a normal key field whose data the program should interpret as character data. The entries in columns 12 and 16 show the location of the key field in the input file. The GSORT program compares the zone and digit portions of each character in positions 3 through 5 of the input records to determine its position in the output file. An optional comment begins in column 40.

In the second and third Field Selection specifications, the fields are identified as data fields in the output file that contains character data. The entries in columns 9 through 16 specify the location of the fields in the input record to be shown as a data field in the output file. Optional comments for these fields begin at column 40.

## Sorting a File with Conditional Include and Omit

The next three example programs use the input file and records described below. In each program, the input record is remapped for the output record.

### Input Records

1	2	3	4	5	6	7
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
first	last name	customer	item	quantity	status	
name		number	number	ordered	of order	
Henry	Molloy		222	4444	333W	
James	Meehan		111	3333	222S	
Nancy	Jones		333	5555	222W	
Don	Meehan		444	5555	111S	
Iris	Johnson		555	5555	111W	

The status of an order is either shipped (S) or waiting to be shipped (W).

**Example: Conditional Include Program**

```

1           2           3           4           5           6
123456789012345678901234567890123456789012345678901234567890
$GSORT
HSORTR    20A           0X 39
I C 60 60EQCW           INCLUDE RECS WITH 60..60 = 'W'
FNU 40 49           1ST KEY FLD: ITM NBR 40..49
FNU 50 59           2ND KEY FLD: QTY ORD 50..59
FDC 1 9           1ST DATA FLD: FIRSTNAME 1..9
FDU 30 39           2ND DATA FLD: CUST NBR 30..39
FDU 40 49           3RD DATA FLD: ITM NBR 40..49
FDU 50 59           4TH DATA FLD: QTY ORD 50..59
    
```

This example produces a regular sort (SORTR), in ascending order. Each output record contains 39 characters in four fields: first name, customer number, item number, and quantity ordered. Only records with a status of W are included in the output file. The first key for sorting is item number, and the second is quantity ordered.

The conditional include program produces the following output file:

```

1           2           3           4           5           6
123456789012345678901234567890123456789012345678901234567890
Henry           222           4444           333
Iris            555           5555           111
Nancy           333           5555           222
    
```

**Example: Conditional Omit Program**

```

1           2           3           4           5           6
123456789012345678901234567890123456789012345678901234567890
$GSORT
HSORTR    20A           0X 39
O C 50 59LTC           200           OMIT RECORDS WITH QTY ORD < 200
I           INCLUDE ALL OTHER RECORDS
FNU 40 49           1ST KEY FLD: ITM NBR 40..49
FNU 50 59           2ND KEY FLD: QTY ORD 50..59
FDC 1 9           1ST DATA FLD: FIRSTNAME 1..9
FDU 30 39           2ND DATA FLD: CUST NBR 30..39
FDU 40 49           3RD DATA FLD: ITM NBR 40..49
FDU 50 59           4TH DATA FLD: QTY ORD 50..59
    
```

This example produces a regular sort (SORTR), in ascending order. Each output record contains 39 characters in four fields: the first name, the customer number, the item number, and the quantity ordered. If the quantity ordered is less than 200 items, the record is omitted. All other records are sorted and included in the output file.

The conditional omit program produces the following output file:

## Using GSORT to Code and Run Sort Programs

	1	2	3	4	5	6
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
James	111	3333	222			
Henry	222	4444	333			
Nancy	333	5555	222			

### Example: Mixed Include and Omit Program

	1	2	3	4	5	6
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
\$GSORT						
HSORTR	20A		0X 40			
O C 50	59GTC		500			OMIT RECORDS WITH QTY ORD > 500
I C 60	60EQCS					INCLUDE 'S'HIPPED RECORDS
FNU 40	49					SORT ON ITM NBR 40..49
FDC 60	60					1ST DATA FLD: STATUS FLAG
FDU 40	49					2ND DATA FLD: ITM NBR
FDC 10	29					3RD DATA FLD: LAST NAME
I C 60	60EQCW					INCLUDE 'W'AITING RECORDS
FNC 10	29					SORT ON LAST NAME
FDC 60	60					1ST DATA FLD: STATUS FLAG
FDC 10	29					2ND DATA FLD: LAST NAME
FDC 1	9					3RD DATA FLD: FIRST NAME
FDU 50	59					4TH DATA FLD: QTY ORD

This example produces a regular sort (SORTR), in ascending order. All output records contain 40 characters. There are two types of output records contained in the same output file.

For the first type of output record, all input records with an order quantity greater than 500 are omitted. From the remaining input records, all records with a status of S are sorted by item number and placed in the output file. The output records have three fields: status, item number, and last name.

The second type of output record consists of all input records with a status of W. They are sorted by customer number. The output record has four fields: status, last name, first name, and quantity ordered.

The mixed include and omit program produces the following output file:

	1	2	3	4	5	6
1234567890	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
S	3333	Meehan				
S	5555	Meehan				
WJohnson		Iris		111		
WJones		Nancy		222		
WMolloy		Henry		333		

## Sorting a File by Record Number

If you do not want to duplicate all the input data in the output file, you can produce a file that contains only relative record numbers. In this case, you do not specify any data fields in the Field Selection specifications. You specify only the key fields.

```

      1      2      3      4      5      6      7
12345678901234567890123456789012345678901234567890123456789012
$GSORT
HSORTA    7D          3  N      TAG SORT
I          INCLUDE-ALL SORT
FNU 26 32          CONTROL FIELD IS HOURLY WAGE
    
```

For this example, the format of the input records is as follows:

Record Code	Employee ID	Employee Name	Hourly Wage	FT/PT
1	2 3	7 8	25 26	32 33 36

The Hourly Wage is the key field and you want to sort in descending order.

### Header Specification

The Header specification indicates that this is a record number sort (SORTA) and the key field is 7 positions long. The input records are to be sorted in descending order. The output file will contain only relative record numbers that are 3 characters long. No error message will be displayed if none of the input records meet the specified conditions in the Record Selection specifications. An optional comment begins at column 44.

### Record Selection Specification

Because this program has no Record Selection specification, the program defaults to an include-all sort. In other SORTA programs you can use Record Selection specifications to specify include or omit options.

### Field Selection Specification

In the one Field Selection specification, the key field is identified as a normal key field containing character data. The entries in columns 9 through 16 show the location of the key field in the input record. An optional comment begins at column 40.

## Sorting for Selected Information

```

      1           2           3           4           5           6           7
12345678901234567890123456789012345678901234567890123456789012
00100$GSORT
00500HSORTR      5A          X 23          SORT TO SELECT INFO
01000 C 33 36EQCFFFF          OMIT RECORDS
01500 I C 33 36EQCQQQ          INCLUDE RECORDS
02000 FNU 3 7          CONTROL FIELD IS EMPLOYEE ID
02500 FDU 3 7          DATA FIELD IS EMPLOYEE ID
03000 FDC 8 25          DATA FIELD IS EMPLOYEE NAME
    
```

In this example, you want to sort by employee ID and in ascending order. In your output file you want to show all part time employees by employee ID and name only. The format of the input records is

Record Code	Employee ID	Employee Name	Hourly Wage	FT/PT
1	2 3	7 8	25 26	32 33 36

### Header Specification

A regularly sorted output (SORTR) file, sorted in ascending order, will be generated. There are 5 characters in the field in the input records used in sorting the records. Information in the key field will not be written into the output file. There will be 23 characters in the output records. An optional comment begins at column 44.

### Record Selection Specifications

For this job, you need two Record Selection specifications. The first identifies criteria for omitting records. The second identifies criteria for including records. In both cases, the C in column 8 means compare both the zone and digit portions of the character. Factor 1, the location of the field to be tested, starts in column 33 of the input record and ends in column 36. If the input record has FFFF in columns 33 through 38, the record is omitted. If the input record has QQQQ, it is included.

### Field Selection Specifications

For this job, there are three Field Selection specifications. The first specifies a normal key field whose contents should be compared as whole characters. The entries in columns 12 and 16 show the starting and ending positions of the key field in the input file. The second and third Field Selection specifications are for data fields that contain character data where both the zone and digit portions are compared.

## Sorting by Integer

```

      1      2      3      4      5      6      7
1234567890123456789012345678901234567890123456789012
$GSORT
HSORTR      8A      X 58
I C      1LEC"
FF      2
FNC 1 7
FDC 1 58
    
```

### Header Specification

This is a regular sort where the total length of the key field for either a positive or negative integer is 7 bytes. The sort will be in ascending order. The smallest negative integer comes first; the largest positive integer comes last in the sort sequence. For example, the sorted output would show numbers in this order: -1, 0, 1, 2.

The key field is dropped when data are written into the output file; only the data portion remains. All 58 characters in the input records are written into the output records.

### Record Selection Specifications

Two record types are specified: positive and negative integers. Both record types are to be included in the sort. Therefore, two sets of include specifications are required.

The first Record Selection specification identifies a conditional include. If position 1, the leftmost byte of the 7-byte key field from the input record, is less than or equal to the character constant for a double quotation mark ("), the record is included in the sort. The character constant indicates that the record contains a positive integer.

The second Record Selection specification is an include-all option; all records not previously selected should be included in the sort. Because the first specification selected records with positive integers only, this specification selected the records remaining: those containing negative integers.

### Field Selection Specifications

The Field Selection specifications show that if the input records meet the conditions defined in the conditional option—the contents of position 1 equals the double quotation mark (") character—the GSORT program should force a 2 into the first position of the key field.

### An Alternate Method of Sorting by Integer

If the Field Selection specifications had been the following, all the data in positions 1 through 58 would have been written into the output file:

## Using GSORT to Code and Run Sort Programs

---

```
      1      2      3      4      5      6      7
12345678901234567890123456789012345678901234567890123456789012
$GSORT
HSORTR      8A      X 58
I C      1LEC"
FF      1
FNC 1 7
FDC 1 58
```

The first Field Selection specification indicates that the character 1 will be forced into the first position of the work record for each record with a negative integer. All records with a negative integer will be forced in front of all records with positive integers when the records are sorted.

The next specification defines the key field as a 7-character field in positions 1 through 7 of the input records; the last Field Selection specification defines data fields that are written into the output file.

## GSORT System Error Messages

A run-time error is either an invalid entry or an invalid combination of two or more entries. Run-time error messages for GSORT jobs are displayed on the terminal.

### **BOTH INPUT AND INPUT1 FILE STATEMENTS PRESENT**

*Cause:* You have specified both an INPUT and INPUT1 file in the same file statement. Either one can be specified, but not both of them.

*Response:* Specify either INPUT or INPUT1, INPUT2, INPUT3, and so on, for multiple input files.

### **SORTA INVALID WITH MULTIPLE INPUT FILES**

*Cause:* You have specified more than one input file when performing a sort for relative record numbers or addresses.

*Response:* Specify only one file for the sort.

### **NO RECORDS INCLUDED: 0=CONTINUE, 3=QUIT**

*Cause:* You have not included any records to be sorted.

## Using GSORT to Code and Run Sort Programs

---

*Response:* Using the command ?<mix number> AX, enter a 0 (zero) to continue the job or a 3 to stop the program.

*Note:* The entries 0 and 3 are GSORT standards.





# Appendix F

## Using the Standard Collating Sequence

Collating sequences are logical sequences used to sort items of data into a particular order. A collating sequence is used by all sort programs to compare characters in the input records to determine whether one character is equal to, greater than, or less than another character.

The standard EBCDIC collating sequence is an ordering of the EBCDIC character set by bit representation. Variations occur in the standard collating sequence, depending on which of the following you use:

- Both the zone and digit portions of characters used in the compare operation
- Only the zone portions of characters in compare operations
- Only the digit portions of characters in compare operations

In most Sort and GSORT jobs, both the zone and digit portions are compared when using the CHARACTER data type. Just the zone or digit is compared in cases in which only the zone or digit of a byte contains valid data.

*Note: When the Sort program compares just the zone or the digit portion of characters, different characters can be evaluated as equals; more than one character can have an identical zone portion or digit portion. For example, the zone portion of the letters a, b, c, d, e, f, g, h, and i are considered equal. The digit portion of the letters b, k, s, B, K, and S are considered equal. However, two different characters cannot have both identical zone and identical digit portions.*

### Comparing Both the Zone and Digit Portions

Table F-1 shows the standard EBCDIC collating sequence used for comparing the entire character, both the zone and digit portions. If you are comparing entire characters, look at the Character column. If you are comparing either zone or digit, look at the appropriate digit in the Hex column shown in Tables F-2 and F-3.

## Using the Standard Collating Sequence

**Table F-1. EBCDIC Collating Sequence for Comparing Both the Zone and Digit Portions of a Character**

Order	Char.	Hex	Order	Char.	Hex
1	0	40	48	-	A1
2	1	4A	49	s	A2
3	.	4B	50	t	A3
4	<	4C	51	u	A4
5	(	4D	52	v	A5
6	+	4E	53	w	A6
7		4F	54	x	A7
8	&	50	55	y	A8
9	!	5A	56	z	A9
10	\$	5B	57	{	C0
11	*	5C	58	A	C1
12	)	5D	59	B	C2
13	;	5E	60	C	C3
14	-	5F	61	D	C4
15	- (minus sign)	60	62	E	C5
16	/	61	63	F	C6
17		6A	64	G	C7
18	, (comma)	6B	65	H	C8
19	%	6C	66	I	C9
20	_ (underscore)	6D	67	}	D0
21	>	6E	68	J	D1
22	?	6F	69	K	D2
23	\	79	70	L	D3
24	:	7A	71	M	D4
25	#	7B	72	N	D5
26	@	7C	73	O	D6
27	' (apostrophe)	7D	74	P	D7
28	=	7E	75	Q	D8

continued

Table F-1. EBCDIC Collating Sequence for Comparing Both the Zone and Digit Portions of a Character (cont.)

Order	Char.	Hex	Order	Char.	Hex
29	!	7F	76	R	D9
30	a	81	77	\	E0
31	b	82	78	S	E2
32	c	83	79	T	E3
33	d	84	80	U	E4
34	e	85	81	V	E5
35	f	86	82	W	E6
36	g	87	83	X	E7
37	h	88	84	Y	E8
38	i	89	85	Z	E9
39	j	91	86	0	F0
40	k	92	87	1	F1
41	l	93	88	2	F2
42	m	94	89	3	F3
43	n	95	90	4	F4
44	o	96	91	5	F5
45	p	97	92	6	F6
46	q	98	93	7	F7
47	r	99	94	8	F8
			95	9	F9

## Comparing Only the Zone Portion

Table F-2 shows the standard EBCDIC collating sequence used for comparing only the zone portions of characters.

**Note:** When several characters share the same position in the sequence, they are considered equal. For example, if you are using only the zone portions of characters, the letters a, b, c, d, e, f, g, h, and i (position 5) are considered equal.

## Using the Standard Collating Sequence

**Table F-2. EBCDIC Collating Sequence for Comparing Only the Zone Portion of a Character**

Order in Sequence	Character	Hexadecimal Representation
1	ø	4A
	.	4B
	<	4C
	(	4D
	+	4E
		4F
	2	!
\$		5B
*		5C
)		5D
;		5E
-		5F
3	/	61
		6A
	, (comma)	6B
	%	6C
	_ (underscore)	6D
	>	6E
	?	6F
4	\	79
	:	7A
	#	7B
	@	7C
	' (apostrophe)	7D
	=	7E
	"	7F
5	a	81

continued

Table F-2. EBCDIC Collating Sequence for Comparing Only the Zone Portion of a Character (cont.)

Order in Sequence	Character	Hexadecimal Representation
	b	82
	c	83
	d	84
	e	85
	f	86
	g	87
	h	88
	i	89
6	j	91
	k	92
	l	93
	m	94
	n	95
	o	96
	p	97
	q	98
	r	99
7	` (tilde)	A1
	s	A2
	t	A3
	u	A4
	v	A5
	w	A6
	x	A7
	y	A8
	z	A9
8	&	50

continued

## Using the Standard Collating Sequence

**Table F-2. EBCDIC Collating Sequence for Comparing Only the Zone Portion of a Character (cont.)**

Order in Sequence	Character	Hexadecimal Representation
	{	C0
	A	C1
	B	C2
	C	C3
	D	C4
	E	C5
	F	C6
	G	C7
	H	C8
	I	C9
9	- (minus sign)	60
	}	D0
	J	D1
	K	D2
	L	D3
	M	D4
	N	D5
	O	D6
	P	D7
	Q	D8
	R	D9
10	\	E1
	S	E2
	T	E3
	U	E4
	V	E5
	W	E6

continued

Table F-2. EBCDIC Collating Sequence for Comparing Only the Zone Portion of a Character (cont.)

Order in Sequence	Character	Hexadecimal Representation
	X	E7
	Y	E8
	Z	E9
11	␣	40
	0	F0
	1	F1
	2	F2
	3	F3
	4	F4
	5	F5
	6	F6
	7	F7
	8	F8
	9	F9

## Comparing Only the Digit Portion

Table F-3 shows the standard EBCDIC collating sequence used for comparing only the digit portion of characters.

*Note:* When several characters share the same position in the sequence, they are considered equal. For example, if you are using only the digit portions of characters, the letters b, k, s, B, K, S, and the numeric 2 (position 3) are all considered equal.



## Using the Standard Collating Sequence

**Table F-3. EBCDIC Collating Sequence for Comparing Only the Digit Portion of a Character**

Order in Sequence	Character	Hexadecimal Representation
1	ø	40
	&	50
	- (minus sign)	60
	{	C0
	}	D0
	0	F0
2	/	61
	a	81
	j	91
	~ (tilde)	A1
	A	C1
	J	D1
	\	E1
	1	F1
3	b	82
	k	92
	s	A2
	B	C2
	K	D2
	S	E2
	2	F2
4	c	83
	l	93
	t	A3
	C	C3
	L	D3
	T	E3
	3	F3

continued

Table F-3. EBCDIC Collating Sequence for Comparing Only the Digit Portion of a Character (cont.)

Order in Sequence	Character	Hexadecimal Representation
5	d	84
	m	94
	u	A4
	D	C4
	M	D4
	U	E4
	4	F4
6	e	85
	n	95
	v	A5
	E	C5
	N	D5
	V	E5
	5	F5
7	f	86
	o	96
	w	A6
	F	C6
	O	D6
	W	E6
	6	F6
8	g	87
	p	97
	x	A7
	G	C7
	P	D7

continued

## Using the Standard Collating Sequence

**Table F-3. EBCDIC Collating Sequence for Comparing Only the Digit Portion of a Character (cont.)**

Order in Sequence	Character	Hexadecimal Representation
	X	E7
	7	F7
9	h	88
	q	98
	y	A8
	H	C8
	Q	D8
	Y	E8
	8	F8
10	\	79
	i	89
	r	99
	z	A9
	l	C9
	R	D9
	Z	E9
	9	F9
11	¢	4A
	!	5A
		6A
	:	7A
12	.	4B
	\$	5B
	, (comma)	6B
	#	7B
13	<	4C

continued

Table F-3. EBCDIC Collating Sequence for Comparing Only the Digit Portion of a Character (cont.)

Order in Sequence	Character	Hexadecimal Representation
	*	5C
	%	6C
	@	7C
14	(	4D
	)	5D
	_ (underscore)	6D
	' (apostrophe)	7D
15	+	4E
	;	5E
	>	6E
	=	7E
16		4F
	- (hyphen)	5F
	?	6F
	"	7F



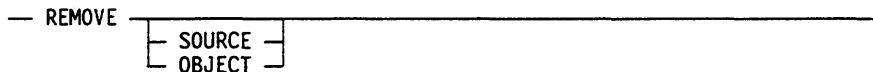
# Appendix G

## Understanding Railroad Diagrams

### What Are Railroad Diagrams?

Railroad diagrams are diagrams that show you the rules for putting words and symbols together into commands and statements that the computer can understand. These diagrams consist of a series of paths that show the allowable structure, constants, and variables for a command or a statement. Paths show the order in which the command or statement is constructed. Paths are represented by horizontal and vertical lines. Many railroad diagrams have a number of different paths you can take to get to the end of the diagram.

### Example



If you follow this railroad diagram from left to right, you will discover three acceptable commands. These commands are

REMOVE

REMOVE SOURCE

REMOVE OBJECT

If all railroad diagrams were this simple, this explanation could end here. However, because the allowed ways of communicating with the computer can be complex, railroad diagrams sometimes must also be complex.

Regardless of the level of complexity, all railroad diagrams are visual representations of commands and statements. Railroad diagrams are intended to

- Show the mandatory items
- Show the user-selected items
- Present the order in which the items must appear
- Show the number of times an item can be repeated
- Show the necessary punctuation

To familiarize you with railroad diagrams, this explanation describes the elements of the diagrams and provides examples.

## Understanding Railroad Diagrams

---

Some of the actual railroad diagrams you will encounter might be more complex. However, all railroad diagrams, simple or complex, follow the same basic rules. They all consist of paths that represent the allowable structure, constants, and variables for commands and statements.

By following railroad diagrams, it is easy to understand the correct syntax for commands and statements. Once you become proficient in the use of railroad notation, the diagrams serve as quick references to the commands and statements.

### Constants and Variables

A constant is an item that cannot be altered. You must enter the constant as it appears in the diagram, either in full or as an allowable abbreviation. If a constant is partially underlined, you can abbreviate the constant by entering only the underlined letters. In addition to the underlined letters, any of the remaining letters can be entered. If no part of the constant is underlined, the constant cannot be abbreviated. Constants can be recognized by the fact that they are never enclosed in angle brackets (< >) and are in uppercase letters.

A variable is an item that represents data. You can replace the variable with data that meets the requirements of the particular command or statement. When replacing a variable with data, you must follow the rules defined for the particular command or statement. Variables appear in railroad diagrams enclosed in angle brackets (< >).

In the following example, BEGIN and END are constants while <statement list> is a variable. The constant BEGIN can be abbreviated since it is partially underlined. Valid abbreviations for BEGIN are BE, BEG, and BEGI.

— BEGIN —<statement list>— END —————|

### Constraints

Constraints are used in a railroad diagram to control progression through the diagram. Constraints consist of symbols and unique railroad diagram line paths. They include

- Vertical bars
- Percent signs
- Right arrows
- Required items
- User-selected items
- Loops
- Bridges

A description of each item follows.

### Vertical Bar

The vertical bar symbol (|) represents the end of a railroad diagram and indicates the command or statement can be followed by another command or statement.

— SECONDWORD — ( —<arithmetic expression>— ) —————|

## Percent Sign

The percent sign (%) represents the end of a railroad diagram and indicates the command or statement must be on a line by itself.

— STOP —————|%

## Right Arrow

The right arrow symbol (>) is used when the railroad diagram is too long to fit on one line and must continue on the next. A right arrow appears at the end of the first line and another right arrow appears at the beginning of the next line.

— SCALERIGHT — ( —<arithmetic expression>— , —————→  
→<arithmetic expression>— ) —————|

## Required Items

A required item can be either a constant, a variable, or punctuation. A required item appears as a single entry, by itself or with other items, on a horizontal line. Required items can also exist on horizontal lines within alternate paths or nested (lower-level) diagrams. If the path you are following contains a required item, you must enter the item in the command or statement; the required item cannot be omitted.

In the following example, the word EVENT is a required constant and <identifier> is a required variable:

— EVENT —<identifier>—————|

## User-Selected Items

User-selected items appear one below the other in a vertical list. You can choose any one of the items from the list. If the list also contains an empty path (solid line), none of the choices are required. A user-selected item can be either a constant, a variable, or punctuation. In the following railroad diagram, either the plus sign (+) or minus sign (-) can be entered before the required variable <arithmetic expression>, or the symbols can be disregarded because the diagram also contains an empty path.

— [ + - ] —<arithmetic expression>—————|

## Loop

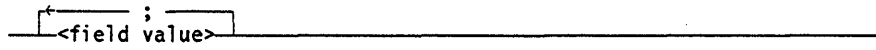
A loop represents an item or group of items that you can repeat. A loop can span all or part of a railroad diagram. It always consists of at least two horizontal lines, one below the other, connected on both sides by vertical lines. The top line is a right-to-left path that contains information about repeating the loop.



## Understanding Railroad Diagrams

---

Some loops include a return character. A return character is a character (often a comma or semicolon) required before each repetition of a loop. If there is no return character, the items must be separated by one or more blank spaces.



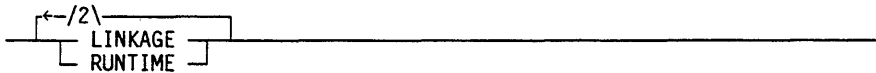
### Bridge

Sometimes a loop also includes a bridge, which is used to show the maximum number of times the loop can be repeated. The bridge can precede the contents of the loop, or it can precede the return character (if any) on the upper line of the loop.

The bridge determines the number of times you can cross that point in the diagram. The bridge is an integer enclosed in sloping lines (/ \). Not all loops have bridges. Those that do not can be repeated any number of times until all valid entries have been used.

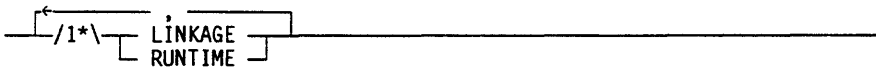


or



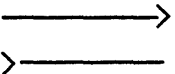
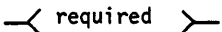
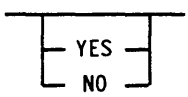
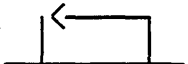
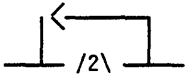


In the first bridge example, you can enter LINKAGE or RUNTIME no more than two times. In the second bridge example, you can enter LINKAGE or RUNTIME no more than three times.

In some bridges an asterisk follows the number. The asterisk means that you must select one item from the group.



The following figure shows the types of constraints used in railroad diagrams.

SYMBOL/PATH	EXPLANATION
	Vertical bar. Indicates that the command or statement can be followed by another command or statement.
	Percent sign. Indicates that the command or statement must be on a line by itself.
	Right arrow. Indicates that the diagram occupies more than one line.
	Required items. Indicates the constants, variables, and punctuation that must be entered in a command or statement.
	User-selected items. Indicates the items that appear one below the other in a vertical list. You select which item or items to include.
	A loop. Indicates an item or group of items that can be repeated.
	A bridge. Indicates the maximum number of times a loop can be repeated.

**Figure G-1. Railroad Constraints**

### Following the Paths of a Railroad Diagram

The paths of a railroad diagram lead you through the command or statement from beginning to end. Some railroad diagrams have only one path, while others have several alternate paths. The following railroad diagram indicates there is only one path that requires the constant LINKAGE and the variable <linkage mnemonic>:

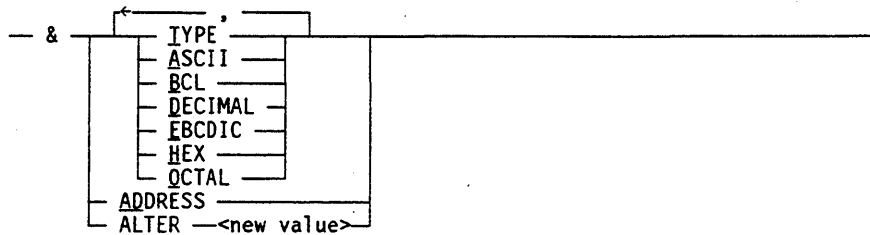
— LINKAGE —<linkage mnemonic>—————|

Alternate paths provide choices in the construction of commands and statements. Alternate paths are provided by loops, user selected items, or a combination of both. More complex railroad diagrams can consist of many alternate paths, or nested (lower-level) diagrams, that show a further level of detail.

For example, the following railroad diagram consists of a top path and two alternate paths. The top path includes an ampersand (&) and the constants (that are

## Understanding Railroad Diagrams

user-selected items) in the vertical list. These constants are within a loop that can be repeated any number of times until all options have been selected. The first alternate path requires the ampersand (&) and the required constant ADDRESS. The second alternate path requires the ampersand (&) followed by the required constant ALTER and the required variable <new value>.



### Railroad Diagram Examples

The following examples show five railroad diagrams and possible command and statement constructions based on the paths of these diagrams.

#### Example 1

<lock statement>

— LOCK — ( — <file identifier> — ) —————|

#### Sample Input

LOCK (F1)

LOCK (FILE4)

#### Explanation

LOCK is a constant and cannot be altered. Because no part of the word is underlined, the entire word must be entered. The parentheses are required punctuation and F1 and FILE4 are sample <file identifier> s.

#### Example 2

<open statement>

— OPEN — [ INQUIRY ] [ UPDATE ] <data base name> —————|

#### Sample Input

OPEN DATABASE1

OPEN INQUIRY DATABASE1

OPEN UPDATE DATABASE1

**Explanation**

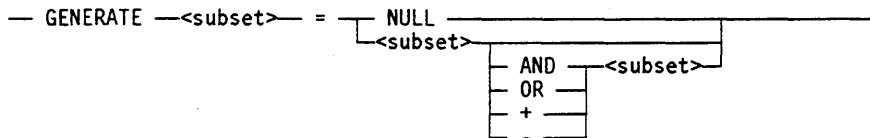
The first sample input shows the constant OPEN followed by the variable DATABASE1, which is a database name. The railroad diagram shows two user-selected items, INQUIRY and UPDATE. However, because there is an empty path (solid line), these entries are not required.

The second sample input shows the constant OPEN followed by the user-selected constant INQUIRY and the variable DATABASE1.

The third sample input shows the constant OPEN followed by the user-selected constant UPDATE and the variable DATABASE1.

**Example 3**

<generate statement>



**Sample Input**

GENERATE Z = NULL

GENERATE Z = X

GENERATE Z = X AND B

GENERATE Z = X + B

**Explanation**

The first sample input shows the GENERATE constant followed by the variable Z, an equal sign, and the user-selected constant NULL.

The second sample input shows the GENERATE constant followed by the variable Z, an equal sign, and the user-selected variable X.

The third sample input shows the GENERATE constant followed by the variable Z, an equal sign, the user-selected variable X, the AND command (from the list of user-selected items in the nested path), and a third variable, B.

The fourth sample input shows the GENERATE constant followed by the variable Z, an equal sign, the user-selectable variable X, the plus sign (from the list of user-selected items in the nested path), and a third variable, B.

## Example 4

<entity reference declaration>

```

— ENTITY REFERENCE —> <entity ref ID> ( <class ID> )
    
```

## Sample Input

ENTITY REFERENCE ADVISOR1 (INSTRUCTOR)

ENTITY REFERENCE ADVISOR1 (INSTRUCTOR), ADVISOR2 (ASST\_INSTRUCTOR)

## Explanation

The first sample input shows the required item ENTITY REFERENCE followed by the variable ADVISOR1 and the variable INSTRUCTOR. The parentheses are required.

The second sample input illustrates the use of a loop by showing the same input as in the first sample followed by a comma, the variable ADVISOR2, and the variable ASST\_INSTRUCTOR. The parentheses are required.

## Example 5

```

— PS — MODIFY —>
    <request number> , <request number>
    <request number> - <request number>
    ALL
    EXCEPTIONS
    <file attribute phrase>
    <print modifier phrase>
    
```

## Sample Input

PS MODIFY 11159

PS MODIFY 11159,11160,11163

PS MODIFY 11159-11161 DESTINATION = "LP7"

PS MOD ALL EXCEPTIONS

## Explanation

The first sample input shows the constants PS and MODIFY followed by the variable 11159, which is a <request number>.

The second sample input illustrates the use of a loop by showing the same input as in the first sample followed by a comma, the variable 11160, another comma, and the final variable 11163.

The third sample input shows the constants PS and MODIFY followed by the user-selected variables 11159-11161, which are <request number>s, and the user-selected variable DESTINATION = "LP7", which is a <file attribute phrase>.

The fourth sample input shows the constants PS and MODIFY followed by the user-selected constant ALL, followed by the user-selected constant EXCEPTIONS. Note that in this sample input, the constant MODIFY has been abbreviated.



# Glossary

In this glossary, definitions taken from outside sources are preceded by an abbreviation enclosed in parentheses. Definitions from the *Dictionary of Computing* are preceded by (DOC).

## A

### **address**

The identification of a location in storage (memory).

### **address couple**

A representation of the address of an item in a program. An address couple consists of two numbers: the first number is a lexical level, and the second number is a displacement (offset) within that lexical level.

### **ADDRROUT file**

In RPG, a record-address file produced by a Sort program or procedure. Each record in the ADDRROUT file is a binary integer, which is the relative record number (zero-relative) of a record in its corresponding data file.

### **ALGOL**

Algorithmic Language. A structured, high-level programming language that provides the basis for the stack architecture of the Unisys A Series systems. ALGOL was the first block-structured language developed in the 1960s and served as a basis for such languages as Pascal and Ada. It is still used extensively on A Series systems, primarily for systems programming.

### **ALPHA**

In the A Series SORT, a data type that includes the letters of the alphabet (A through Z and a through z) and special characters. In B Series SORT, ALPHA also includes the numerals 0 through 9.

### **alphanumeric data**

The letters of the alphabet (A through Z and a through z), special characters, and the numerals 0 through 9.

### **alternate collating sequence**

A user-defined collating sequence that causes records or characters to be arranged in an order different from the order permitted by the standard collating sequence.

### **ascending order**

An arrangement of items in which the order progresses consecutively from the lowest-valued item to the highest-valued item.

## B

### **BCL**

See Burroughs Common Language.



## Glossary

---

### bit

The most basic unit of computer information. The word *bit* is a contraction of *binary digit*. A bit can have one of two values: binary 0 (sometimes referred to as OFF) and binary 1 (sometimes referred to as ON).

### Boolean

Pertaining to variables, data items, and attributes having a value of TRUE or FALSE.

### Burroughs Common Language (BCL)

An obsolete code using 6-bit character representation.

### byte

On Unisys A Series systems, a measurable group of eight consecutive bits having a single usage. In data communications, a byte is often referred to as a character or an octet.

## C

### CANDE

See Command and Edit.

### CARD file

In RPG, the first file that the compiler reads for program data.

### CCR

See compiler control record.

### COBOL

Common Business-Oriented Language. A widely used, procedure-oriented language intended for use in solving problems in business data processing. The main characteristics of COBOL are the easy readability of programs and a considerable degree of machine independence. COBOL is the most widely used procedure-oriented language.

### code file

See object code file.

### collating sequence

(DOC) A set of rules establishing the order in which items will be arranged in a set. Common collating sequences are alphabetic order and numerical order with, often, additional rules for dealing with symbols, punctuation, and spaces.

### Command and Edit (CANDE)

A time-sharing Message Control System (MCS) that allows a user to create and edit files, and develop, test, and execute programs, interactively.

### compiler

A computer program that translates instructions written in a source language, such as COBOL or ALGOL, into machine-executable object code.

### compiler control option

An individual compiler directive that appears in a compiler control record (CCR). Compiler control options are also referred to as compiler dollar options or dollar options.

**compiler control record (CCR)**

A record in a source program that begins with a dollar sign (\$) and contains one or more options that control various compiler functions. These specifications can appear anywhere in the source program unless otherwise specified. The term compiler control image (CCI) is a nonpreferred synonym.

**constant**

An object whose value is assigned during program compilation and cannot be changed during program execution.

**control field**

A field that identifies the relationship of a record to other records.

**D****data field**

An area in a data record that contains one particular piece of information.

**data item**

An element of data.

**descending order**

An arrangement of items in which the order progresses consecutively from the largest item to the smallest item.

**digit**

(DOC) A graphic character that represents an integer; for example, any of the characters 0 through 9 in the decimal system.

**digit portion**

The low order, or least significant, bits in a byte.

**disk file header**

A data structure that contains information about a disk file, such as the physical location of the file on the disk and various file attributes. A disk file header is also referred to as a header.

**displacement**

(DOC) A numerical difference between two values, one of which is a base or reference value. In a Sort or GSORT program, the value that specifies the position of the first, or most significant, element in the field.

**double-precision**

Pertaining to an arithmetic value that is represented internally as a signed-magnitude mantissa and exponent and is contained in two words.

### E

#### **EBCDIC**

Extended Binary Coded Decimal Interchange Code. An 8-bit code representing 256 graphic and control characters that are the native character set of most mainframe systems.

#### **Editor**

A Unisys utility program designed to create and modify program source and data files.

### F

#### **family**

The name of the disk or disk pack on which a physical file is located.

#### **Field Selection Specification**

A format used to define the control and data fields for a GSORT job.

#### **file attributes**

Elements that describe characteristics of a file and provide information the system needs to handle the file. Examples of file attributes are the file title, record size, number of areas, and date of creation. For disk files, permanent file attribute values are stored in the disk file header.

In the SORT language, file attributes can be specified only in the FILE statements.

#### **floating-point literal**

The value of a literal shown with the decimal part of a number multiplied by the power of 10 and used as an alternative for a standard numeric literal.

### H

#### **Header Specification**

A format used to indicate the type of GSORT job to be performed and some of the sort parameters.

#### **hex**

See hexadecimal.

#### **hexadecimal (hex)**

Pertaining to the base 16 numbering system. Decimal digits 0 through 9 are represented by the characters 0 through 9. Decimal digits 10 through 15 are represented by the characters A through F.

#### **hexadecimal character**

One of a set of characters that includes 0 (zero) through 9 and A through F used to represent quantities in base 16. The characters A through F represent the decimal values 10 through 15, respectively.

**hexadecimal literal**

(1) In RPG, an item used in the formation of alphanumeric characters. It can be used to form a bit pattern that cannot otherwise be represented in the RPG source image. (2) In COBOL, a character-string bounded by at signs (@). The string of characters must consist of one or more characters chosen from the set of hexadecimal characters.

**I****include set**

A combination of a Header specification, Record Selection specifications, and Field Selection specifications that indicates types of records to be sorted. Contrast with *omit set*.

**index**

A value used to specify a particular element of an array variable.

**input record boundary**

The beginning or end of an input record. For Sort files, records begin in column 1 and end in column 90. Sort language statements are in columns 1 through 72.

**integer**

A whole number.

**Integrated Tape and Disk (ITD)**

A procedure that allows input files for a SORT job to come from both tape and disk sources.

**internal file name**

The name used to declare a logical file in a program. The internal name of a file is given by the value of its INTNAME file attribute. Work Flow Language (WFL) file equation statements can reference the file by implicitly or explicitly specifying an INTNAME value that matches the INTNAME attribute of a file in a program.

**ITD**

See Integrated Tape and Disk.

**K****key**

A field in a record that is used to sort a file.

**key field**

A field that identifies the relationship of a record to other records and determines the order of records in a sorted file. A key field is also used to locate or identify a record.

**keyword**

In programming languages, a reserved word that must be present when the format in which the word appears is used in a source program.

## L

### literal

A character string enclosed by a pair of special characters whose value is implied by the ordered set of characters that compose the string.

## M

### Master Control Program (MCP)

An operating system on A Series systems. The MCP controls the operational environment of the system by performing job selection, memory management, peripheral management, virtual memory management, and dynamic subroutine linkage.

### MAXRECSIZE

A file attribute that gives the maximum size, in frames, of records in a logical file. For port files, MAXRECSIZE specifies the maximum text size for all subfiles in the port file.

### MCP

See Master Control Program.

### MCS

See Message Control System.

### Message Control System (MCS)

A program that controls the flow of messages between terminals, application programs, and the operating system. MCS functions can include message routing, access control, audit and recovery, system management, and message formatting.

### Message Translation Utility

A software tool for translating compiled program output messages from one natural language to one or more others.

### MLS

See MultiLingual System.

### MultiLingual System (MLS)

A system for developing and accessing output messages, online help text, and menu screens in different natural languages, such as English, French, and Spanish.

## N

### NEWSOURCE file

The output source file of the Pascal, RPG, or SORT language compiler that is either a copy of the CARD input (if the MERGE option is FALSE) or the result of merging the CARD and SOURCE files (if the MERGE option is TRUE).

---

## O

**object code file**

A file produced by a compiler when a program is compiled successfully. The file contains instructions in machine-executable object code.

**omit set**

A combination of a Header specification, Record Selection specifications, and Field Selection specifications that indicates types of records not to be sorted. Contrast with *include set*.

**operand**

An entity on which operations are performed. Operands are grouped with operators (such as +, AND, and OR) to create expressions.

## P

**packed decimal format**

A format in which each byte in a field represents two numeric digits. If the format is right-signed, the rightmost byte contains a digit and a positive or negative sign. If the format is left-signed, the leftmost byte contains a digit and a positive or negative sign.

**parity bit**

A bit appended to an array of bits to force the sum of all bits in the array to be odd (for odd parity) or even (for even parity), as required by the convention established.

## R

**real**

Pertaining to signed or unsigned, fractional or whole values in single-precision, floating-point form.

**record**

A group of logically related items of data in a file that are treated as a unit.

**Record Selection Specification**

A format used to indicate the items to include or delete in a GSORT job.

**relational operator**

A reserved word, a relational character, a group of consecutive reserved words, or a group of consecutive reserved words and relational characters used in the construction of a relation condition.

**RPG**

Report Program Generator. A high-level, commercially oriented programming language used most frequently to produce reports based on information derived from data files.

**run-time error**

An error occurring during the execution of a program, which causes the system software to terminate execution of that program abnormally.

### S

**single-precision**

An arithmetic value that is represented internally as a signed-magnitude mantissa and exponent and is contained in one word.

**SORT language**

A language that allows the user to write programs for sorting and merging files on the A Series system.

**SOURCE file**

The secondary input file from which the compiler reads previously stored source images.

**source program**

A program coded in a language that must be translated into machine language before execution. The translator program is usually a compiler.

**stack**

A region of memory used to store data items in a particular order, usually on a last-in, first-out basis.

**string**

A connected sequence or group of characters.

**summary data field**

When the GSORT option of Sort is specified, a data field used for accumulated totals in summary (SORTRS) jobs.

**symbolic file**

A file that contains a source program.

**syntax error**

An error that occurs when the rules or grammar of a language is violated.

### T

**tag sort**

A type of sort that uses only the key and the address of each record. The data records remain in place until the final merge of the tag begins. At that time, the addresses contained within the tags are used to retrieve the records in the correct sequence. Also known as a record address sort.

**task attribute**

Any of a number of items that describe and control various aspects of the execution of a process.

**task variable**

An object that is used to interrogate or modify the task attributes of a particular process.

**U****unpacked decimal format**

A format in which numbers are represented by both their zone and digit. For each character in an unpacked data field, a numeric value is placed in the digit portion. A positive or negative sign is placed in the zone position of the rightmost character in the data field.

**V****version information field**

The field that consists of columns 81 through 90 of SORT language records.

**W****WFL**

See Work Flow Language.

**word**

A unit of computer memory. On A Series systems, a word consists of 48 bits used for storage, plus tag bits used to indicate how the word is interpreted.

**work file**

A file that the user accesses using the Command and Edit (CANDE) GET command or creates using the CANDE *MAKE* command. All editing commands entered through CANDE can make changes only to the current work file.

**Work Flow Language (WFL)**

A Unisys language used for constructing jobs that compile or run programs on A Series systems. WFL includes variables, expressions, and flow-of-control statements that offer the programmer a wide range of capabilities with regard to task control.





# Bibliography

- A Series CANDE Operations Reference Manual* (form 8600 1500). Unisys Corporation. |
- A Series Editor Operations Guide* (form 8600 0551). Unisys Corporation. |
- A Series File Attributes Programming Reference Manual* (form 8600 0064). Unisys Corporation. Formerly *A Series I/O Subsystem Programming Reference Manual*. |
- A Series MultiLingual System (MLS) Administration, Operations, and Programming Guide* (form 8600 0288). Unisys Corporation. |
- A Series System Software Installation Guide, Volume 2: System Initialization* (form 1170263). Unisys Corporation.
- A Series System Software Support Reference Manual* (form 8600 0478). Unisys Corporation. |
- A Series System Software Utilities Operations Reference Manual* (form 8600 0460). Unisys Corporation. |
- A Series Work Flow Language (WFL) Programming Reference Manual* (form 8600 1047). Unisys Corporation.
- Dictionary of Computing*. Frank J. Galland (ed.). New York: John Wiley and Sons, 1982. |



# Index

## A

- A Series, converting to, D-1
  - acceptable syntax, 5-11
  - accepted, ignored statements, D-1
  - changes required in file statement, D-1
  - comments, handling of, D-1
  - equating file names, examples of, D-6
  - statements not supported, list of, D-1
- accumulated totals, E-7
- accumulated totals sort, (See SORTRS)
- actual record size, D-3
  - in B 1000 FILE statement, conversion of, D-3
- adding a new character to a key or data field in GSORT job, E-39
- address couple
  - definition of, Glossary-1
  - inclusion of, in listing, 5-8
- address, definition of, Glossary-1
- ADDRROUT file
  - definition of, Glossary-1
  - effect on conversion, D-10
  - in GSORT job, E-7
- ADDRROUT sort, (See SORTA)
- ALGOL, definition of, Glossary-1
- allocation of memory, default, 4-7
- ALPHA
  - data type
    - in B 1000 statements, conversion of, D-7
    - internal representation of, 2-8
    - use of, 2-6
  - definition of, Glossary-1
  - hexadecimal literal conversion, associating, 2-12
  - list of valid literal types, 2-12
- ALPHA NUMERIC, 2-6
  - data type
    - internal representation of, 2-8
    - hexadecimal literal conversion, associating, 2-12
    - integer literal conversion, associating, 2-12
    - list of valid literal types, 2-12
  - alphanumeric data
    - comparing and interpreting, E-25
    - defining in B 1000 statements, D-7
    - definition of, Glossary-1
  - alternate collating sequence
    - coding Field Selection for normal or opposite key fields, E-41
    - coding Header record, E-12
    - definition of, Glossary-1
    - differences from standard collating sequence, E-3
    - for entire key field, E-16
      - effect on factor 1 and factor 2, E-28
    - for specified key fields, E-17
    - making characters equal, E-17
    - using ALTSEQ statement to define, E-14
    - with GSORT option, 5-7
- ALTSEQ statement, E-4, E-14
  - coding, E-15
  - ending the statement, E-15
  - for an entire key field, E-16
  - for specified key fields, E-17
  - identification of an ALTSEQ statement, E-15
  - identifying records in the statement, E-15
  - making characters equal, E-17
  - purpose and order of, E-4
  - relationship to alternate collating sequence, E-14
  - relationship to Header record, E-12
  - specifying hexadecimal equivalents, E-16
  - table of entries, E-15
  - use of hexadecimal when shifting to a different order
    - zone and digit comparison, E-14
- AND relationship
  - for binary values, E-26

## Index

---

- for include or omit option of Record Selection, E-21
  - area, disk, D-5
  - area, records per, (See records)
  - < areas >, D-5
  - ascending order
    - definition in GSORT job, E-7
    - definition in Sort job, 3-4
    - definition of, Glossary-1
    - designating in key field, 3-4
    - selecting by key field in GSORT job, E-12
  - attribute
    - for files, (See file attributes)
    - for task, (See task attribute)
- ## B
- < B 1000 file input part >, D-2
  - < B 1000 file name >, D-5
  - < B 1000 file output part >, D-4
  - < B 1000 file statement >, D-2
  - B 1000 data type, declaration of, 2-7
  - < B 1000 data type >
    - declaration of, 2-7
    - defining format of data item, D-7
  - < B 1000 name >, D-6
  - < B 1000 pack >, D-6
  - B 1000 statements
    - accepted but ignored by A Series, D-1
  - B 1000 statements, conversion of, D-1
    - accepted, but ignored, D-1
    - comments, D-1
    - data type, D-7
    - DELETE, D-9
    - FILE, D-1
      - actual record size element, D-3
      - CARD element, D-3
      - DEFAULT element, D-5
      - DISK element, D-2
      - examples of, D-6
      - file input element, D-2
      - file name element, D-5
      - file output element, D-4
      - header information, D-3
      - logical records element, D-3
      - MULTI element, D-3
      - pack name element, D-6
      - PAPER element, D-3
      - records per area element, D-3
      - RELEASE element, D-5
      - removing input file, D-3
      - TAPE element, D-3
      - user name element, D-6
      - VARIABLE element, D-4, D-5
  - INCLUDE, D-9
  - MEMORY, D-9
  - MERGE, D-9
  - NOPRINT, D-1, D-9
  - not supported, D-1
  - RECORDS, D-10
  - SORT, D-10
  - SYNTAX, D-10
    - syntax for, 5-11
  - TAGSORT, D-10
  - TAPESORT, D-11
  - TEACH, D-1
  - WORKPACK, D-11
  - < B 1000 user >, D-6
  - basic syntax components, 2-3
  - BCL, (See Burroughs Common Language, definition of)
  - BIAS statement, conversion of, D-1
  - binary values, as basis of selecting records in GSORT, E-26
  - bit
    - defining key location in, D-7
    - definition of, Glossary-2
    - number, allowable values, 2-5
  - BIT
    - data type
      - internal representation of, 2-8
      - use of, 2-6
    - hexadecimal literal conversion, associating, 2-12
    - list of valid literal types, 2-12
    - syntax of in displacement, 2-4, 2-5
    - < bit number >, 2-5
  - Boolean
    - compiler control record, use of, 5-1
    - definition of, Glossary-2
    - operators, order of evaluation, D-9
    - option in compiler control record, 5-3
  - Boolean compiler control option
    - CODE, 5-4
    - DELETE, 5-4
    - ERRORLIST, 5-6
    - GSORT, 5-6
    - LIST, 5-7
    - LIST\$, 5-8
    - LISTDELETED, 5-7
    - LISTDOLLAR, 5-8
    - LISTP, 5-8
    - MAP, 5-8

MERGE, 5-9  
 NEW, 5-9  
 NEWSEQERR, 5-10  
 REFORMAT, 5-11  
 SEQ, 5-11  
 SEQUENCE, 5-11  
 SINGLE, 5-12  
 Boolean expression of file attribute, 3-3  
 < Boolean file attribute name >, 3-3  
 Burroughs Common Language, definition of,  
 Glossary-2  
 byte  
 defining key location in, D-7  
 definition of, Glossary-2  
 BYTE, syntax of in displacement, 2-4

## C

calculating key field length in GSORT job,  
 E-11  
 calculating output record length in GSORT  
 job, E-13  
 CANDE, (See Command and Edit)  
 CARD  
 file  
 definition of, Glossary-2  
 description of, B-1  
 restricting read to, 5-9  
 in B 1000 FILE statement, conversion of,  
 D-3  
 category 1 keywords, A-1  
 category 2 keywords, A-2  
 category 3 keywords, A-1  
 CCL, (See compiler control record)  
 CCR, (See compiler control record)  
 CHARACTER data type, use of, 2-7  
 character data, as EBCDIC characters, E-25  
 CLEAR compiler control option, use of, 5-4  
 < clear option >, 5-4  
 closing tape with B 1000 FILE RELEASE  
 option, D-5  
 COBOL, definition of, Glossary-2  
 CODE compiler control option, use of, 5-4  
 code file, (See object code file)  
 CODE file  
 description of, B-2  
 suppression, method of, D-10  
 < code option >, 5-4  
 code, producing listing of object file, 5-4  
 coding  
 ALTSEQ statement, E-15

Field Selection specifications, E-30  
 GSORT program, E-4  
 Header specifications, E-8  
 negative unpacked numbers, E-29  
 Record Selection specifications, E-18  
 coding differences between Sort and GSORT  
 programs, E-4  
 collate, (See NO COLLATE data type, use  
 of)  
 COLLATE B 1000 statement, not supported  
 by A Series, D-1  
 COLLATE processing statement, 4-1  
 < collate statement >, 4-1  
 collating sequence  
 alternate sequence with GSORT option,  
 5-7  
 definition of, Glossary-2  
 in ALTSEQ statement, E-14  
 in Header record, E-12  
 colon, use of to denote comments, 2-2  
 Command and Edit, 1-1  
 defaults for  
 ERRORLIMIT compiler control option,  
 5-5  
 GSORT compiler control option, 5-6  
 LIST compiler control option, 5-7  
 definition of, Glossary-2  
 initiating  
 compilation from, 1-2  
 syntax-only compilations from, 1-3  
 use of  
 COMPILE command, 1-3  
 COMPILE SYNTAX command, 1-3  
 RUN command, 1-3  
 use of commands, 1-1  
 comments  
 converting B 1000 to A Series, D-1  
 in Field Selection record, E-33, E-42  
 in Header record, E-14  
 in Record Selection record, E-21, E-30  
 in WFL, 1-5  
 including in Sort program, 2-2  
 COMP data type, use of, 2-6  
 compare character, in Field Selection record,  
 E-39  
 compare constant, in Record Selection record,  
 E-28  
 compare date keyword, in Record Selection  
 record, E-30  
 compare field name, in Record Selection  
 record, E-28

## Index

---

- compare field, in Record Selection record, E-28
  - compare keyword, in Record Selection record, E-28
  - compare literal, in Record Selection record, E-29
  - comparing characters
    - alphanumeric data, E-25
    - digit only, E-36
    - specifying digit only, E-25
    - specifying zone and digit, E-25
    - specifying zone only, E-25
    - zone and digit, E-36
    - zone only, E-36
  - for a forced key field, E-39
  - for a summary overflow indicator field, E-39
  - numeric data, E-26
    - packed, E-36
    - specifying digit only, E-26
    - specifying zone and digit, E-26
    - specifying zone only, E-26
    - unpacked, E-36
  - signed, E-36
  - table of digit only comparison, F-7
  - table of zone and digit comparison, F-1
  - table of zone only comparison, F-3
  - unsigned, E-36
- COMPILE command, use of, 1-3
- COMPILE SYNTAX command, use of, 1-3
- compile time, defining file attributes in WFL, 1-5
- compiler control image, (See compiler control record)
- compiler control option
  - available, list of, 1-5
  - definition of, Glossary-2
  - use of, 5-4
    - CLEAR immediate option, 5-4
    - CODE Boolean option, 5-4
    - DELETE Boolean option, 5-4
    - ERRORLIMIT value option, 5-5
    - ERRORLIST Boolean option, 5-6
    - GSORT Boolean option, 5-6
    - LIST Boolean option, 5-7
    - LIST\$ Boolean option, 5-8
    - LISTDELETED Boolean option, 5-7
    - LISTDOLLAR Boolean option, 5-8
    - LISTP Boolean option, 5-8
    - MAP Boolean option, 5-8
    - MERGE Boolean option, 5-9
    - NEW Boolean option, 5-9
    - NEWSEQERR Boolean option, 5-10
    - REFORMAT Boolean option, 5-11
    - SEQ Boolean option, 5-11
    - Sequence Base value option, 5-12
    - SEQUENCE Boolean option, 5-11
    - Sequence Increment value option, 5-12
    - SINGLE Boolean option, 5-12
    - VERSION value option, 5-12
    - \$MERGE in record format, 2-2
    - \$SEQUENCE in record format, 2-2- compiler control record
  - definition of, Glossary-3
  - designator (\$), use of, 5-3
  - list of options, 1-5
  - position in Sort program, 2-2
  - producing listing of temporary, 5-8
  - submitting through TASKSTRING attribute, 5-2
  - syntax of, 5-2, 5-3
  - types of, 5-1
  - use of, 5-1
    - Boolean option, 5-1, 5-3
    - immediate option, 5-1, 5-3
    - permanent type, 5-1
    - temporary type, 5-1
    - value option, 5-1, 5-3
  - use of POP statement, 5-4
  - use of RESET statement, 5-4
  - use of SET statement, 5-4
- < compiler control records >, 5-3
- compiler, definition of, Glossary-2
- compiling, 1-2
  - combined compile and run, example of, 1-3
  - effect of syntax errors, 1-3
  - explanation of, 1-2
  - files of type
    - SEQUENTIAL, 1-3
    - TEXT, 1-3
  - for immediate execution, 1-3
  - for syntax-only, 1-3
    - example of, 1-3
  - from CANDE, 1-3
  - in WFL, 1-4
    - example of, 1-4
    - for immediate execution, 1-5
  - separate compile and run, example of, 1-3
- condition for record selection, 4-12
- < condition >, 4-12
- conditional forced field, use of, E-34
- conditional include, use of, E-20
- conditionally forcing a character into a key field, E-39

constant, definition of, Glossary-3  
 continuation entry, in Record Selection  
   record, E-21  
 control field, definition of, Glossary-3  
 control file, description of, C-1  
 conversion  
   effect of ADDRROUT files, D-10  
   of B 1000 statements not supported by  
     A Series, D-1  
   of hex literal, 2-12  
   of integer literal, 2-12  
   of literal, 2-11  
 converting B 1000 programs, D-1  
   accepted, but ignored statements, D-1  
   comments, handling of, D-1  
   statements not supported, list of, D-1  
 creating Sort program  
   example of, 1-2  
   using CANDE, 1-1  
 cycle number, assigning, 5-13  
 < cycle number >, 5-13

## D

D, as double-precision designator in a real  
 literal, 2-11  
 D, for deleted source-language input records  
 on listing, 5-7  
 data field  
   adding a new character, E-39  
   definition of, Glossary-3  
   types  
     forced, E-36  
     normal, E-35  
     summary, E-35  
 data field types, E-35  
   maximum lengths, E-40  
 data interpretation, in Record Selection  
 record, E-25  
 data item, definition of, Glossary-3  
 data length designation, 2-9  
 data type  
   in A Series, use of  
     ALPHA, 2-6  
     ALPHA NUMERIC, 2-6  
     BIT, 2-6  
     CHARACTER, 2-7  
     COMP, 2-6  
     DECIMAL, 2-7  
     DIGIT, 2-7  
     DISPLAY, 2-6

  DISPLAY NUMERIC, 2-6  
   DOUBLE, 2-6  
   HEX, 2-7  
   INTEGER, 2-5  
   IS, 2-7  
   IS CHARACTER, 2-7  
   NO COLLATE, 2-6  
   PACKED, 2-6  
   REAL, 2-5  
   sign clause, 2-6  
   ZONE, 2-7  
   in B 1000 declarations, 2-7  
   in B 1000 statements, D-8  
   in converting B 1000 statements, D-7  
   in Field Selection record, E-36  
   internal representation, table of, 2-8  
 DECIMAL data type, use of, 2-7  
 decimal number  
   packed, E-37  
   unpacked, E-36  
 DEFAULT in B 1000 FILE statement,  
   conversion of, D-3  
 defining a character as a summary overflow  
 indicator, E-39  
 defining field type, E-33  
 DELETE  
   as record selection statement, 4-11  
   B 1000 statement, conversion of, D-9  
   Boolean compiler control option, use of,  
     5-5  
   < delete option >, 5-4  
   < delete statement >, (See < include and  
     delete statement > )  
 deleted records, method of listing, 5-7  
 descending order  
   definition in GSORT job, E-7  
   definition in Sort job, 3-4  
   designating in key field, 3-4  
   selecting by key field in GSORT job, E-12  
 destination of error messages, control of, 5-6  
 detecting summary data overflow, E-41  
 determining order input records are written  
   to output file, E-11  
 differences of GSORT and Sort programs,  
   E-2, E-4  
 digit, 2-3  
   definition of, Glossary-3  
   syntax of, 2-3  
   use of with E or D designators, 2-11  
 DIGIT  
   data type



- in B 1000 statements, conversion of, D-8
- internal representation of, 2-8
  - use of, 2-7
- defining as displacement units, 2-4
- field, associating with literal, 2-11
- list of valid literal types, 2-12
- digit only comparison
  - for binary values, E-26
  - specifying for alphanumeric data, E-25
  - specifying for numeric data, E-26
  - table of, F-7
- digit portion, definition of, Glossary-3
- < digit >, 2-3
- disk area, designating for output file, D-5
- disk file
  - altering sequence numbers, in, 5-10
  - containing sort input, generating, 5-9
  - header, definition of, Glossary-3
- disk pack assignment, use of, 4-10
- disk sort
  - associated syntax errors, 4-2
  - default memory allocation, 4-7
  - requesting, 4-2
- DISK, in B 1000 FILE statement, conversion of, D-2
- DISKANDTAPE processing statement, 4-2
  - < diskandtape statement >, 4-2
- DISK file, description of, C-1
- DISKF file, description of, C-1
- DISKSORT processing statement, 4-2
  - < disksort statement >, 4-2
  - < disp-1 >, 4-13
  - < disp-2 >, 4-13
- displacement
  - causing run-time error, 2-4
  - definition of, Glossary-3
  - in conditions, 4-13
  - in key fields, 3-4
  - syntax for, 2-4
    - BIT units, 2-5
    - BYTE units, 2-4
    - DIGIT units, 2-4
    - HEX units, 2-4
    - PACKED units, 2-4
    - WORD units, 2-4
  - use of, 2-3
  - < displacement >, 2-4, 3-4
- DISPLAY data type, use of, 2-6
- DISPLAY NUMERIC data type, use of, 2-6
- dispositions allowed in WFL job, 1-4
- dollar sign, 5-3

- in listing, meaning of, 5-8
- DOUBLE
  - data type
    - internal representation of, 2-8
    - use of, 2-6
  - list of valid literal types, 2-12
- double-precision
  - definition of, Glossary-3
  - designation of real literal, 2-11
- DUPCHECK B 1000 statement, not supported by A Series, D-1

## E

- E, as single-precision designator in a real literal, 2-11
- EBCDIC collating sequence
  - comparing digit only, F-8
  - comparing zone and digit, F-1
  - comparing zone only, F-4
- EBCDIC, definition of, Glossary-4
- Editor, definition of, Glossary-4
- efficiency, increasing
  - using GSORT compiler control option, 5-6
  - using sort parameter statements, 4-7
- ending an ALTSEQ statement, E-15
- equal key field
  - in GSORT program, E-11
  - in Sort program, 4-4
  - when determining length of GSORT output record, E-13
- equal records, treatment of, 4-4
- error
  - effect of parity error on sort, 4-8
  - in syntax associated with
    - DISKANDTAPE processing statement, 4-2
    - DISKSORT processing statement, 4-2
    - INVALID CHARACTER, 1-1
    - invalid literal types, 2-12
    - literal and DIGIT, 2-11
    - literal and ZONE, 2-11
    - MEMORY parameter statement, 4-8
    - MEMORYONLY processing statement, 4-3
    - MERGE processing statement, 4-3
    - PARITY DISCARD parameter statement, 4-8
    - RECORDS parameter statement, 4-9
    - STABLE processing statement, 4-5

- TAGSEARCH parameter statement, 4-9
  - TAGSORT processing statement, 4-5
  - TAPESORT processing statement, 4-6
  - WORKFAMILY parameter statement, 4-10
  - WORKSIZE parameter statement, 4-10
  - in syntax, effect on compilation, 1-3
  - run-time
    - caused by displacement size, 2-4
    - for GSORT jobs, E-50
  - error handling, in GSORT, E-5
  - error message
    - destination, control of, 5-6
    - file used for, B-1
    - for no records selected in GSORT job, E-14
  - ERRORFILE file
    - description of, B-2
    - destination of error messages, 5-6
  - ERRORLIMIT compiler control option, use of, 5-5
    - <errorlimit option>, 5-5
  - ERRORLIST compiler control option, use of, 5-6
    - <errorlist option>, 5-6
  - ERRORS file
    - description of, B-2
    - destination of error messages, 5-6
  - executing a GSORT program, E-5
  - executing differences between Sort and GSORT programs, E-4, E-5
  - Extended Binary Coded Decimal Interchange Code, (See EBCDIC, definition of)
- F**
- factor 1, E-26
    - noting length in Record Selection record, E-27
    - noting location in Record Selection record, E-26
    - relationship to factor 2, E-27
    - use in alternate collating sequence, E-28
  - factor 2, E-28
    - as a constant, E-29
    - as a date keyword, E-30
    - as an signed numeric constant, E-29
    - as an unsigned alphanumeric constant, E-29
  - noting location in Record Selection record, E-28
  - relationship to factor 1, E-27
  - use in alternate collating sequence, E-28
  - use in comparing with factor 1, E-28
  - family
    - definition of, Glossary-4
    - designation of, 4-10
  - family name, 4-10
    - <family name>, 4-10
  - field
    - DIGIT, associating with literal, 2-11
    - key, (See key field)
    - ZONE, associating with literal, 2-11
  - field location, noting in input record, E-38
  - Field Selection record, (See Field Selection specification)
  - Field Selection specification, E-4, E-30
    - alternate collating sequence, E-41
    - coding, E-30
    - comments, E-42
    - compare character, E-39
    - definition of, Glossary-4
    - field location, E-38
    - field type, E-33
    - fixed format, E-31
    - forced character, E-39
    - identification of a Field Selection record, E-32
    - overflow field length, E-40
    - purpose and order of, E-4
    - relationship to include sets, E-23
    - relationship to omit sets, E-25
    - remaps, E-40
    - sequence number, E-32
    - table of entries, E-31
  - FIELD statement, 3-3
    - <field statement>, 3-3
  - field type, in Field Selection record, E-33
  - file
    - ADDROUT
      - effect on conversion process, D-10
      - altering sequence numbers, in, 5-10
    - CARD, description of, B-1
    - CODE, description of, B-2
    - disk, (See disk file)
    - error messages, description of, B-1
    - IN, description of, C-1
    - input, (See input file)
      - designation of, 1-4
      - requesting removal of, D-3
    - LINE, description of, B-1

- list of those used by SORT compiler, B-1
  - NEWSOURCE
    - description of, B-1
  - organization required for input files, 3-3
  - output
    - designating maximum disk area, D-5
    - designation of, 1-4
  - requesting identical parameters in input and output files, D-5
  - secondary source, description of, B-1
  - SOURCE
    - description of, B-1
  - symbolic
    - name of, B-1
  - title designation in
    - MERGE compiler control option, 5-9
    - NEW compiler control option, 5-10
  - type.
    - defining, 1-2
    - SEQUENTIAL use of COMPILE command, 1-3
    - TEXT use of COMPILE command, 1-3
  - used as
    - control file, C-1
    - first input file, C-1
    - object code file, B-2
    - output file, C-1
    - secondary input file, C-1
    - sort programs, C-1
    - work file, C-1
  - file attribute equation, use of, 1-4
  - file attributes, 3-3
    - defaults, C-2
    - defining in WFL
      - at run time, 1-5
      - during compilation, 1-5
    - definition of, Glossary-4
    - designation of, example, 3-3
    - expressed as Boolean, 3-3
    - expressed as string, 3-3
    - FILE ORGANIZATION, input file restriction, 3-3
    - FILE statement, use in, 3-2
    - FILETYPE, effect of, C-1
    - in FILE statement, 3-1
    - syntax of, 3-2
    - < file attributes >, 3-2
  - FILE B 1000 statement
    - VARIABLE element, conversion of, D-4
  - FILE B 1000 statement, VARIABLE element, conversion of, D-5
  - file name, D-6
  - B 1000 FILE statement
    - conversion of, D-5
    - syntax of, D-5
    - equating B 1000 to A Series, examples of, D-6
  - FILE statement
    - changes required when converting, D-1
    - description, 3-1
    - file attributes, use of, 3-2
    - input file, 3-1
    - output file, 3-1
    - PURGE option, 3-2
    - syntax, 3-1
    - tape release, 3-2
    - with GSORT, E-5
    - < file statement >, 3-1
    - < file title >, 5-10
  - FILEORGANIZATION attribute, 3-3
  - FILETYPE attribute, effect of, C-1
  - first input file, description of, C-1
  - fixed format
    - of ALTSEQ statement, E-15
    - of Field Selection specification, E-31
    - of Header specification, E-9
    - of Record Selection specification, E-18
  - floating-point literal, definition of, Glossary-4
  - force-all field, use of, E-35
  - forced character, in Field Selection record, E-39
  - forced data field
    - options, E-37
    - use of, E-36
  - forced key field, E-34
    - comparing characters, E-39
    - conditional, E-34
    - force-all, E-34
    - unconditional, E-34
- ## G
- GET command in CANDE, use of, 1-1
  - GO code file disposition, use of, 1-4
  - graphic literal, use of, 2-10
    - < graphic literal >, 2-10
  - GSORT compiler control option
    - coding a GSORT program, E-4
    - controlling results, E-36
    - default value, 5-6
    - error handling, E-5
    - executing a program, E-5
    - identifying include or omit action, E-20

key field types, E-33  
 list of additional sort features, 5-6  
 major key field, E-33  
 minor key field, E-33  
 preventing summary data overflow, E-41  
 program examples  
   producing same result as Sort program,  
   E-1  
   SORTA, E-47  
   with include, E-44  
   with include and omit, E-44  
   with omit, E-44  
   with record selection, E-43, E-48  
   with record selection and by integer,  
   E-49  
   without record selection, E-42  
 purpose and order of specifications and  
 statement, E-4  
 run-time error messages, E-50  
 running a GSORT job, E-5  
 selecting the fields in an output record,  
 E-8  
 sorting order, E-7  
 specifications  
   ALTSEQ statement, E-14  
   Field Selection, E-30  
   Header, E-8  
   Record Selection, E-18  
 syntax of, 5-7  
 use of, 5-6  
 < gsort option >, 5-6  
 GSORT programs  
   coding differences from Sort programs,  
   E-4  
   differences from Sort program, E-2  
   running differences from Sort programs,  
   E-5  
   similarities to Sort program, E-1  
 GSORT specifications  
   ALTSEQ statement, E-14  
   Field Selection, E-30  
   Header, E-8  
   Record Selection, E-18  
 guidelines for mixing include and omit  
 options, E-21

## H

header information, obtaining from B 1000  
   FILE statement, D-3  
 Header record, (See Header specification)

Header specification, E-4, E-8  
   coding, E-8  
   collating sequence, E-12  
   comments, E-14  
   definition of, Glossary-4  
   equal key field, E-11  
   fixed format, E-9  
   identical key field, E-11  
   identification of Header record, E-10  
   key field length, E-11  
   null output, E-14  
   output option, E-13  
   output record length, E-13  
   print option, E-13  
   purpose and order of, E-4  
   record-sorting order, E-12  
   sequence number, E-10  
   table of entries, E-9  
   type of job, E-11  
 hex, (See hexadecimal)  
 HEX data type, (See hexadecimal)  
 < hex digit >, 2-10  
 < hex literal >, 2-10  
 hexadecimal, 2-7  
   definition of, Glossary-4  
   internal representation of, 2-8  
   list of valid literal types, 2-12  
   literal, 2-10  
   literal conversion, 2-12  
   syntax of  
     in displacement, 2-4  
     in literal, 2-10  
 hexadecimal character, definition of,  
   Glossary-4  
 hexadecimal literal, definition of, Glossary-5

## I

identical key field, in Header record, E-11  
 identifying a record  
   as a Field Selection specification, E-32  
   as a Header specification, E-10  
   as an ALTSEQ statement, E-15  
   as an include Record Selection  
   specification, E-20  
   as an omit Record Selection specification,  
   E-20  
 identifying the type of sorted output, in  
   Header record, E-11  
 image, source-language, (See  
   source-language image)

- immediate compiler control option
    - CLEAR, 5-4
    - use of, 5-1, 5-3
  - <immediate option>, 5-3
  - IN files, use of, C-1
  - INCLUDE
    - as record selection statement, 4-11
    - B 1000 statement, conversion of, D-9
    - <include and delete statement>, 4-11
  - include option, E-20
    - AND relationship, E-21
    - conditional include, E-20
    - in Record Selection record, E-20
    - include-all, E-20
    - relationship to Field Selection record, E-23
    - table of include sets, E-21
    - use with omit option, E-20
  - include set, definition of, Glossary-5
  - <include statement>, (See <include and delete statement>)
  - include-all, use of, E-20
  - index
    - definition of, Glossary-5
    - designating length in TAGSORT processing statement, 4-6
  - index type, designating, 4-6
  - indexing record locations, 4-5
  - INPLACE B 1000 statement, not supported by A Series, D-1
  - input file
    - designating for Sort program, 1-4
    - designation of, 1-4
    - in FILE statement, 3-1, E-5
    - processing, use of, 3-4
    - repeating parameters in output file, D-5
    - requesting removal of, in B 1000 FILE statement, D-3
    - required organization, 3-3
  - input from both tape and disk sources, allowing, 4-2
  - <input part>, 3-1
  - input record boundary, definition of, Glossary-5
  - input record format
    - for GSORT compiler control option, 5-6
    - for Sort program, 2-2
  - input statement, file used for, B-1
  - integer, 2-3
    - compared to integer literal, 2-3
    - definition of, Glossary-5
    - literal
      - conversion of, 2-12
      - use of, 2-11
    - not splitting across input record boundary, 2-2
    - syntax of, 2-3
      - in literal, 2-10
  - INTEGER
    - data type
      - internal representation of, 2-8
      - use of, 2-5
    - hexadecimal literal conversion, associating, 2-12
    - list of valid literal types, 2-12
  - integer expression of file attribute, 3-3
  - <integer file attribute name>, 3-3
  - integer literal
    - compared to integer, 2-3
    - <integer literal>, 2-10
    - <integer>, 2-3
  - Integrated Tape and Disk, 4-2, (See also DISKANDTAPE processing statement)
    - definition of, Glossary-5
    - sort
      - designating number of tapes, 4-2
      - example of, 4-2
      - syntax errors, associated with, 4-2
      - syntax of, 4-2
  - internal file name, definition of, Glossary-5
  - internal representation of data types, 2-8
  - internationalization collating sequence, 4-1, E-3, E-9, E-12
  - INVALID CHARACTER associated syntax error, 1-1
  - IS CHARACTER data type, use of, 2-7
  - IS data type, use of, 2-7
  - ITD, (See Integrated Tape and Disk, Integrated Tape and Disk)
- J**
- job disposition, valid, 1-4
- K**
- key
    - and data field storage, table of, 2-9
    - defining elements in, 2-3
    - definition of, Glossary-5
    - length designation, 2-9

location defining in  
 bits for B 1000 statements, D-7  
 bytes for B 1000 statements, D-7

key field  
 adding or replacing a character, E-39  
 ascending order designation, 3-4  
 conditionally forcing a character, E-39  
 defining position of first element, 3-4  
 definition of, Glossary-5  
 descending order designation, 3-4  
 designation, example of, 3-5  
 determining presence in output file, E-13  
 displacement, use of, 3-4  
 equal  
 in GSORT program, E-11  
 in Sort program, 4-4  
 identical, E-11  
 in GSORT, E-33  
 length  
 in GSORT program, E-11  
 in Sort program, 3-4  
 order, use of, 3-4  
 replacing more than one character, E-40  
 type designation, 3-5  
 types in GSORT job  
 conditional forced, E-34  
 force-all, E-35  
 forced, E-33  
 major, E-33  
 minor, E-33  
 normal, E-33  
 opposite, E-33  
 unconditional forced, E-34  
 use of, 3-4

key field length, in Header record, E-11  
 < key field >, 3-4

KEY statement, 3-3  
 < key statement >, 3-3

keyword  
 definition of, Glossary-5  
 LIBRARY in WFL, use of, 1-4  
 not splitting across input record boundary,  
 2-2  
 recognized by GSORT compiler, list of,  
 E-30  
 recognized by SORT compiler, list of, A-1

SORT  
 in WFL, use of, 1-4  
 use of, 1-2

words  
 beginning statements, list of, A-1  
 occurring in statements, list of, A-2

## L

LEADING SEPARATE sign positioning, use  
 of, 2-8

LEADING sign positioning, use of, 2-8

LEADING ZONE sign positioning, use of, 2-8

LEFT SEPARATE sign positioning, use of,  
 2-8

LEFT sign positioning, use of, 2-8

LEFT ZONE sign positioning, use of, 2-8

length, 2-9  
 comparing in Boolean conditions, 4-13  
 designating, 2-9  
 in key fields  
 GSORT program, E-11  
 Sort program, 3-4  
 of index designating in TAGSORT  
 processing statement, 4-6  
 < length >, 2-9, 3-4

LIBRARY  
 code file disposition, use of, 1-4  
 keyword in WFL, use of, 1-4

LIBRARY GO code file disposition, use of,  
 1-4

limiting input files with MERGE processing  
 statement, 4-3

LINE file  
 description of, B-1  
 destination of error messages, 5-6

LIST compiler control option, 5-7  
 file used with, B-1  
 replacing B 1000 NOPRINT statement,  
 D-1  
 < list option >, 5-7

LIST\$ option, (See LISTDOLLAR compiler  
 control option)

LISTDELETED compiler control option, 5-7  
 < listdeleted option >, 5-7

LISTDOLLAR compiler control option, 5-8  
 < listdollar option >, 5-8

listing  
 of object code, producing, 5-4  
 primary input file records, producing, 5-8  
 production of  
 single-spaced, 5-12  
 temporary CCRs, 5-8  
 to include address couples, 5-8  
 source-language input records, method of,  
 5-7

LISTP compiler control option, 5-8  
 < listp option >, 5-8

literal

- allowable types, table of, 2-12
- conversion, 2-11
- conversion of hexadecimal, 2-12
- conversion of integer, 2-12
- definition of, Glossary-6
- graphic, (See graphic literal, use of)
- in DIGIT fields, 2-11
- in ZONE fields, 2-11
- not splitting across input record boundary, 2-2
- real
  - designation of double-precision, 2-11
  - designation of single-precision, 2-11
- syntax, 2-10
- use of, 2-9
  - hexadecimal, 2-10
  - integer, 2-11
  - real, 2-11
- < literal >, 2-9, 2-11
- Local Data Specification, use of, 1-4
- logical records in B 1000 FILE statement, conversion of, D-3
- lowercase characters, use of, 1-1

## M

- major key field, E-33
- MAKE command in CANDE, use of, 1-1
- making characters equal, ALTSEQ statement, E-17
- MAP compiler control option, 5-8
- < map option >, 5-8
- Master Control Program
  - definition of, Glossary-6
  - SORT procedure, use of, 1-2
- maximum block size, syntax in B 1000 FILE statement, D-2
- MAXRECSIZE
  - calculation of, 3-2
  - definition of, Glossary-6
- MCP, (See Master Control Program)
- MCS, (See Message Control System, definition of)
- MEMORY
  - B 1000 statement, conversion of, D-9
- memory allocation, default, 4-7
- MEMORY parameter statement, 4-7
- memory size, designating, 4-8
- < memory statement >, 4-8
- memory-only sorts, use of, 4-3
- MEMORYONLY processing statement, 4-3

- < memoryonly statement >, 4-3
- MERGE
  - A Series processing statement, 4-3
  - B 1000 statement, conversion of, D-9
  - compiler control option, 2-2, 5-9
    - syntax of, 5-9
- < merge option >, 5-9
- < merge statement >, 4-4
- merging multiple input files
  - in GSORT programs
    - selecting records, E-18
    - use of equal key field ordering, E-11
  - in Sort programs, 1-1
    - selecting records, 4-11
    - use of DELETE option, 5-5
    - use of SEQUENCE option, 5-11
- Message Control System, definition of, Glossary-6
- Message Translation Utility, definition of, Glossary-6
- messages
  - error when no records selected, E-14
  - printed, E-13
  - run-time error, E-50
- messages file, error, (See error message)
- minor key field, E-33
- mixing include and omit options
  - guidelines, E-21
- MLS, (See MultiLingual System)
- mnemonic expression of file attribute, 3-3
- modifying Sort programs using CANDE, 1-1
- MULTI in B 1000 FILE statement, conversion of, D-3
- MultiLingual System
  - definition of, Glossary-6
  - for translation of messages, 1-1

## N

- name of files, B 1000, D-5
- name of pack, in B 1000 FILE statement, D-6
- name of statement, in ALTSEQ statement, E-15
- names of user in B 1000 FILE statement, D-6
- NC data type in B 1000 statements, conversion of, D-7
- NEW compiler control option, 5-9
  - file used with, B-1
  - syntax of, 5-10
  - < new option >, 5-9

- NEWSEQERR compiler control option, 5-10  
 <newseqerr option>, 5-10
- NEWSOURCE file  
 definition of, Glossary-6  
 description of, B-1
- NO COLLATE data type, use of, 2-6
- NOPRINT B 1000 statement, D-1  
 conversion of, D-9
- normal data fields, use of, E-35
- normal key field, E-34  
 alternate collating sequence, E-17
- noting location of fields in input record, E-38
- null output, in Header record, E-14
- number sign, purpose of, 1-2
- numbers, sequence, (See sequence number)
- NUMERIC, 2-6  
 data type in B 1000 statements, conversion of, D-7
- numeric data  
 comparing, E-25, E-26  
 defining in B 1000 statements, D-7  
 interpreting, E-25  
 signed, (See signed)
- O**
- object code file  
 definition of, Glossary-7  
 description of, B-2  
 disposition of, 1-4
- object code, producing listing of, 5-4
- offset-2, use of, 2-4  
 <offset-2>, 2-4
- offset, use of, 2-4  
 <offset>, 2-4, (See also displacement)
- omit option  
 in Record Selection record, E-20  
 OR relationship, E-21  
 relationship to Field Selection record, E-25  
 table of omit sets, E-24  
 use with include option, E-20
- omit set, definition of, Glossary-7
- operand, definition of, Glossary-7
- operator  
 evaluation order  
 comparing B 1000 and A Series, D-9
- opposite key field, E-34  
 alternate collating sequence, E-17
- optimization  
 using GSORT compiler control option, 5-6  
 using RECORDS parameter statement, 4-9
- option  
 list of compiler control, 1-5  
 use of compiler control, 5-4  
 CLEAR, 5-4  
 CODE, 5-4  
 DELETE, 5-4  
 ERRORLIMIT, 5-5  
 ERRORLIST, 5-6  
 GSORT, 5-7  
 LIST, 5-7  
 LIST\$, 5-8  
 LISTDELETED, 5-7  
 LISTDOLLAR, 5-8  
 LISTP, 5-8  
 MAP, 5-8  
 MERGE, 5-9  
 NEW, 5-9  
 NEWSEQERR, 5-10  
 REFORMAT, 5-11  
 SEQ, 5-11  
 SEQUENCE, 5-11  
 Sequence Base, 5-12  
 Sequence Increment, 5-12  
 SINGLE, 5-12  
 VERSION, 5-12
- optional SORT statement, use of, 2-1
- OR relationship, in include/omit option of Record Selection, E-21
- order  
 in key field designating  
 ascending order, 3-4  
 descending order, 3-4  
 of evaluating operators, comparing B 1000 and A Series, D-9
- order input record are written into output file, E-11
- order of GSORT specifications and statement, E-4  
 <order>, 3-4
- organization required for input file, 3-3
- OUT file  
 description of, C-1  
 in FILE statement, 3-1  
 in GSORT job, E-5
- output file  
 associated file attributes, 3-1  
 default attributes, associated with, C-2  
 description of, C-1  
 designating for Sort program, 1-4  
 designating maximum disk area, D-5



## Index

---

- designation of, 1-4
- file statement, definition in, 3-2
- in FILE statement, 3-1, E-5
- requesting identical parameters to input file, D-5
- SORTA, E-7
- sorted, construction of, 4-9
- SORTR, E-6
- SORTRS, E-7
- types for GSORT option, E-6
- output option, in Header record, E-13
- < output part >, 3-1
- output record length, in Header record, E-13
- output records, selecting fields, E-8
- output, producing single-spaced listing, 5-12
- overflow field length, in Field Selection record, E-40
- overflow indicators, E-41
- OVERRIDE B 1000 statement, not supported by A Series, D-1

## P

- pack name, in B 1000 FILE statement, D-6
- PACKED
  - data type
    - internal representation of, 2-8
    - use of, 2-6
  - integer literal conversion, associating, 2-12
  - list of valid literal types, 2-12
  - syntax of in displacement, 2-4
- packed data, specifying, E-26
- packed decimal format, definition of, Glossary-7
- packed decimal number, E-37
- PACKED SIGNED data type, internal representation of, 2-8
- PAPER in B 1000 FILE statement, conversion of, D-3
- parameter statements
  - MEMORY, 4-7
  - PARITY DISCARD, 4-8
  - RECORDS, 4-9
  - TAGSEARCH, 4-9
  - use of, 4-7
  - WORKFAMILY, 4-10
  - WORKSIZE, 4-10
- parity
  - error, effect on sort, 4-8
- parity bit, definition of, Glossary-7
- PARITY DISCARD parameter statement, 4-8
  - < parity discard statement >, 4-8
- patch number, assigning, 5-13
  - < patch number >, assigning, 5-13
- percent sign, using to
  - define comments in WFL, 1-5
  - denote comments, 2-2
- permanent compiler control record, use of, 5-1
  - < pointer file attribute name >, 3-3
- POP statement in compiler control record, use of, 5-4
- position
  - defining most significant, 2-3
  - of sign using
    - LEFT SEPARATE, 2-8
    - LEFT ZONE, 2-8
    - RIGHT SEPARATE, 2-8
    - RIGHT ZONE, 2-7
- precision
  - double
    - definition of, Glossary-3
    - designation of in a real literal, 2-11
  - single
    - designation of in a real literal, 2-11
- preventing summary data overflow, E-41
- primary input file, producing listing of, 5-8
- primary key, use of, 3-4
- print option, in Header record, E-13
- printed messages, specifying types, E-13
- processing statements, 4-1
  - COLLATE, 4-1
  - DISKANDTAPE, 4-2
  - DISKSORT, 4-2
  - MERGE, 4-3
  - STABLE, 4-4
  - TAGSORT, 4-5
  - TAPESORT, 4-6
  - use of
    - MEMORYONLY, 4-3
- program examples, using GSORT options, E-42
- program file, description of, C-1
- PURGE
  - in B 1000 FILE statement, conversion of, D-3
  - option in FILE statement, 3-2
- purposes of GSORT specifications and statement, E-4

**Q**

- question mark, use of in WFL, 1-4
- quotation marks, use of in file designation, 1-4

**R**

- <r.b>, D-3
- Railroad diagrams, explanation of, G-1
- real
  - definition of, Glossary-7
  - literal
    - double-precision designation, 2-11
    - single-precision designation, 2-11
    - syntax of, 2-10
    - use of, 2-11
- REAL
  - data type
    - internal representation of, 2-8
    - use of, 2-5
  - list of valid literal types, 2-12
- <real literal>, 2-10
- record
  - actual size, (See actual record size)
  - compiler control, (See compiler control record)
  - definition of, Glossary-7
  - estimate in RECORDS parameter statement, 4-9
  - format, (See Sort program)
  - input to SORT, creating a disk file of, 5-9
  - method of listing, 5-7
  - number file, use of, 5-7
  - producing listing for primary input file, 5-8
  - size, calculation of, 3-2
  - splitting across boundary, 2-2
  - treatment of equal key values, 4-4
- record address sort, (See tag sort)
- <record estimate statement>, 4-9
- record number estimation, 4-9
- Record Selection record, (See Record Selection specification)
- Record Selection specification, E-4, E-18
  - AND option, E-21
  - coding, E-18
  - comments, E-30
  - compare constant, E-28
  - compare date keyword, E-30
  - compare field, E-28
  - compare field name, E-28
  - compare keyword, E-28
  - compare literal, E-29
  - continuation, E-21
  - data interpretation, E-25
  - definition of, Glossary-7
  - factor 1 location, E-26
  - factor 2 location, E-28
  - fixed format, E-18
  - guidelines for mixing include and omit options, E-21
  - identification of a Record Selection record, E-20
  - include option, E-20
  - include/omit specifications, E-20
  - OR option, E-21
  - purpose and order of, E-4
  - relationship of factor 1 and factor 2, E-27
  - relationship of include sets to Field Selection record, E-23
  - relationship of omit sets to Field Selection record, E-25
  - sequence number, E-20
  - showing relationship between Record Selection specifications, E-21
  - table of entries, E-18
  - table of include sets, E-21
  - table of omit sets, E-24
- record selection statement, in Sort program, 4-11
- <record selection statement>, (See <include and delete statement>)
- record-sorting order, in Header record, E-12
- record, compiler control designator, 5-3
- records
  - logical, (See logical records in B 1000 FILE statement, conversion of per area in B 1000 FILE statement, conversion of, D-3)
- RECORDS
  - A Series Sort parameter statement, 4-9
  - conversion of B 1000 statement, D-10
- REFORMAT compiler control option
  - replacing B 1000 TEACH statement, D-1
  - use of, 5-11
- <reformat option>, 5-11
- regularly sorted output, (See SORTR)
- relation, use of, 4-12
- <relation>, 4-12
- relational operator, definition of, Glossary-7
- relative record locations, requesting, 4-5
- relative record numbers

- use in GSORT, E-7
- use in Sort program, 5-7
- RELEASE
  - conversion of in B 1000 FILE statement, D-5
  - option in A Series FILE statement, 3-2
- release number, assigning, 5-13
- < release number >, 5-13
- release of tape, requesting in file statement, 3-2
- remapping records
  - creating output records with GSORT, E-2
  - program examples, E-44
  - with GSORT compiler control option, E-3
- remaps, in Field Selection record, E-40
- removing input file in B 1000 FILE statement, request for, D-3
- replacing a key field character, in GSORT job, E-39, E-40
- Report Program Generator, (See RPG, definition of)
- required SORT statements, 2-1
- RESET LIST compiler control option, use of, D-9
- RESET statement in compiler control record, 5-4
- RIGHT option, use in sign positioning, 2-7
- RIGHT SEPARATE option, use in sign positioning, 2-8
- RIGHT ZONE option, use in sign positioning, 2-7
- < rpa >, D-3
- RPG, definition of, Glossary-7
- RSA data type in B 1000 statements, conversion of, D-8
- RSN data type in B 1000 statements, conversion of, D-8
- < rsz >, D-3
- RUN command
  - use of, 1-3
  - with GSORT, E-5
- run-time
  - defining file attributes in WFL, 1-5
  - error messages, GSORT, E-50
- run-time error
  - caused by displacement size, 2-4
  - definition of, Glossary-7
- running
  - GSORT program, E-5
  - Sort program, 1-2
  - from WFL, 1-4

## S

- SA data type in B 1000 statements, conversion of, D-8
- SAVE command in CANDE, use of, 1-1
- saving Sort program, example of, 1-2
- secondary input file, description of, C-1
- secondary source file, description of, B-1
- secondary source-language input file, control of, 5-9
- selecting
  - a record-sorting order, in GSORT job, E-12
  - records based on binary values, in GSORT job, E-26
  - records for sorting or merging, in GSORT job, E-18
  - records for sorting or merging, in Sort program, 4-11
  - the fields in an output record, in GSORT job, E-8
- semicolon, use of, 2-2
- SEPARATE sign positioning, use of, 2-8
- separating SORT statements, 2-2
- SEQ compiler control option, use of, 5-11
- Sequence Base compiler control option, use of, 5-12
- < sequence base option >, 5-12
- SEQUENCE compiler control option, 2-2
  - use of, 5-11
- < sequence increment option >, 5-12
- Sequence Increment, compiler control option, use of, 5-12
- sequence number
  - assigning new values, 5-11
  - assigning values to, 5-12
  - in Field Selection record, E-32
  - in Header record, E-10
  - in Record Selection record, E-20
  - method of altering, 5-10
- sequence number field, use of, 2-2
- < sequence option >, 5-11
- SEQUENTIAL file type, compiling, 1-3
- SET statement in compiler control record, 5-4
- sign
  - clause, use of, 2-6
  - positioning of, 2-7
    - LEADING SEPARATE, 2-8
    - LEADING ZONE, 2-8
    - LEFT SEPARATE, 2-8
    - LEFT ZONE, 2-8

- RIGHT SEPARATE, 2-8
- RIGHT ZONE, 2-7
- TRAILING SEPARATE, 2-8
- TRAILING ZONE, 2-7
- < sign clause >, 2-6
- signed
  - alphanumeric data, defining in B 1000
    - statements, D-8
  - numeric data, defining in B 1000
    - statements, D-8
- signed data, E-36
- similarities of GSORT and Sort programs, E-1
- simple condition, use of, 4-12
- < simple condition >, 4-12
- SINGLE compiler control option, use of, 5-12
- < single option >, 5-12
- single-precision
  - definition of, Glossary-8
  - designation of real literal, 2-11
- single-spaced listing, producing, 5-12
- size of
  - record, calculating, 3-2
  - work file, designating, 4-10
- SN data type in B 1000 statements, conversion of, D-8
- SORT B 1000 statement, conversion of, D-10
- SORT compiler files, B-1
  - CARD, B-1
  - CODE, B-2
  - ERRORFILE, B-2
  - ERRORS, B-2
  - LINE, B-1
  - NEWSOURCE, B-1
  - SOURCE, B-1
- SORT compiler, method of record submission, 5-2
- SORT DATA CARD, use of, 1-4
- sort efficiency, increasing, 4-7
- SORT keyword
  - in WFL, 1-4
  - purpose of, 1-2
- SORT language
  - definition of, Glossary-8
  - keywords, list of, A-1
- sort optimization, 4-9
  - using RECORDS parameter statement, 4-9
- SORT parameter statements
  - MEMORY, 4-7
  - PARITY DISCARD, 4-8
  - RECORDS, 4-9
- TAGSEARCH, 4-9
  - use of, 4-7
- WORKFAMILY, 4-10
- WORKSIZE, 4-10
- sort performed in memory, request for, 4-3
- SORT procedure, use of, 1-2
- SORT processing statements
  - COLLATE, 4-1
  - DISKANDTAPE, 4-2
  - DISKSORT, 4-2
  - STABLE, 4-4
  - TAGSORT, 4-5
  - TAPESORT, 4-6
  - use of, 4-1
    - COLLATE, 4-1
    - DISKANDTAPE, 4-2
    - MEMORYONLY, 4-3
    - MERGE, 4-3
- Sort program, 2-1
  - compiling
    - from CANDE, 1-3
    - from WFL, 1-4
  - creating, 1-1
  - differences from GSORT program, E-2
  - example of
    - combined compile and run, 1-3
    - compiling for syntax-only, 1-3
    - creating, 1-2
    - designating input file, 1-4
    - designating output file, 1-4
    - saving, 1-2
    - separate compile and run, 1-3
    - writing, 1-2
  - files, description of, C-1
    - DISKC, C-1
    - DISKF, C-1
    - IN, C-1
    - OUT, C-1
  - including comments, 2-2
  - modifying, 1-1
  - position of compiler control records, 2-2
  - record format, description of, 2-2
  - running
    - from CANDE, 1-3
    - from WFL, 1-4
  - similarities to GSORT program, E-1
  - structure of, 2-1
  - syntax, 2-1
  - syntax-only compilation, 1-3
  - TASKVALUE task attribute, use of, 1-6
  - use of, 1-1
  - writing, 1-1

- use of uppercase characters, 1-1
- < sort program >, 2-1
- SORT statements, 2-1
  - compiler control record, use of, 5-4
  - FIELD, use of, 3-3
  - FILE, use of, 3-1
  - KEY, use of, 3-3
  - numbers for, 1-1
  - optional, 2-1, 4-1
  - parameter statements, 4-7
  - processing statements, 4-1
  - record selection statement, 4-11
  - required, 2-1, 3-1
  - separation of, 2-2
- < sort statements >, 2-1
- sort, requesting use of
  - disk, 4-2
  - tape, 4-6
- sort, tag, (See tag sort)
- SORTA, E-7, E-50
  - order input records written into output file, E-11
  - program example, E-47
  - use of Field Selection records, E-31
- sorted output files
  - construction of in Sort program, 4-9
  - identifying type in Header record, E-11
  - record addresses, (See SORTA)
  - regular, (See SORTR)
  - summary, (See SORTRS)
- sorting order, definition, 3-4, E-7
- sorting with accumulated totals, E-7
- SORTR, E-6
  - content of records, E-7
  - determining presence of key field in output file, E-13
  - order input records written into output file, E-11
  - program example
    - by integer, E-49
    - with both include and omit, E-44
    - with conditional include, E-44
    - with conditional omit, E-44
    - with record selection, E-43, E-48
    - without record selection, E-42
  - selecting fields for output record, E-8
  - specifying length of output record, E-13
  - use of Field Selection records, E-31
  - use of forced data fields, E-36
  - use of normal data fields, E-35
- SORTRS, E-7
  - determining presence of key field in output file, E-13
  - specifying length of output record, E-13
  - use of Field Selection records, E-31
  - use of forced data fields, E-36
  - use of normal data fields, E-35
  - use of summary data fields, E-35
- SOURCE file
  - definition of, Glossary-8
  - description of, B-1
  - source program, definition of, Glossary-8
  - source-language image
    - assigning new sequence numbers, 5-11
    - deletion of, 5-4
  - spacing, producing single-spaced listing, 5-12
  - specification file, printing specifications and messages, B-1
  - specifying
    - hexadecimal equivalents in an ALTSEQ statement, E-16
    - overflow indicators, E-41
  - splitting
    - integer across record boundary, 2-2
    - keyword across record boundary, 2-2
    - literal across record boundary, 2-2
- STABLE processing statement, 4-4
- < stable statement >, 4-5
- stack value, assigning to Boolean option, 5-4
- stack, definition of, Glossary-8
- standard collating sequence
  - coding Header record, E-12
  - use of, F-1
- statement numbers, 1-1, (See also sequence number)
- storage locations, including information in listing, 5-8
- string expression of file attribute, 3-3
- string, definition of, Glossary-8
- structure
  - of GSORT program, E-4
  - of Sort program, 2-1
- summary data, E-7
- summary data field
  - data types, E-38
  - definition of, Glossary-8
  - use of, E-35
- summary data overflow
  - defining indicator, E-39
  - detecting, E-41
  - field length, E-41
  - length of field, E-40
  - preventing, E-41

- summary overflow indicator field
  - comparing characters, E-39
- summary overflow indicator, in Field selection record, E-39
- symbolic file
  - definition of, Glossary-8
  - name of, B-1
- syntax
  - error
    - caused by VARIABLE element of FILE statement, D-4, D-5
- SYNTAX
  - B 1000 statements, conversion of, D-10
  - code file disposition, use of, 1-4
- syntax error
  - associated with DISKANDTAPE processing statement, 4-2
  - associated with DISKSORT processing statement, 4-2
  - associated with INVALID CHARACTER, 1-1
  - associated with invalid literal types, 2-12
  - associated with MEMORY parameter statement, 4-8
  - associated with MEMORYONLY processing statement, 4-3
  - associated with MERGE processing statement, 4-3
  - associated with PARITY DISCARD parameter statement, 4-8
  - associated with RECORDS parameter statement, 4-9
  - associated with STABLE processing statement, 4-5
  - associated with TAGSEARCH parameter statement, 4-9
  - associated with TAGSORT processing statement, 4-5
  - associated with TAPESORT processing statement, 4-6
  - associated with WORKFAMILY parameter statement, 4-10
  - associated with WORKSIZE parameter statement, 4-10
  - associating literal with DIGIT, 2-11
  - associating literal with ZONE, 2-11
  - definition of, Glossary-8
  - effect on compilation, 1-3
  - syntax of
    - ALTSEQ statement, E-14
    - B 1000 FILE statement, D-2
    - data type, D-7
    - file input part, D-2
    - file name element, D-5, D-6
    - maximum block size, D-2
    - output part, D-4
- basic components, 2-3
- BIT in displacement, 2-4, 2-5
- BYTE in displacement, 2-4
- compiler control record, 5-2, 5-3
- DELETE record selection statement, 4-11
- digit, 2-3
- DIGIT in displacement, 2-4
- displacement, 2-4
- Field Selection specification, E-30
- FIELD statement, 3-4
- file attributes, 3-2
- FILE statement, 3-1
- Header specification, E-8
- HEX in displacement, 2-4
- INCLUDE record selection statement, 4-11
- integer, 2-3
- Integrated Tape and Disk sort, 4-2
- KEY statement, 3-4
- literal, 2-10
  - graphic, 2-10
  - hex, 2-10
  - integer, 2-10
  - real, 2-10
- offset, 2-4
- offset-2, 2-4
- PACKED in displacement, 2-4
- parameter statements
  - MEMORY, 4-8
  - PARITY DISCARD, 4-8
  - RECORDS, 4-9
  - TAGSEARCH, 4-9
  - WORKFAMILY, 4-10
  - WORKSIZE, 4-11
- processing statements
  - COLLATE, 4-1
  - DISKANDTAPE, 4-2
  - DISKSORT, 4-2
  - MEMORYONLY, 4-3
  - MERGE, 4-4
  - STABLE, 4-5
  - TAGSORT, 4-6
  - TAPESORT, 4-7
- Record Selection specification, E-18
- record selection statement, 4-11
- Sort program, 2-1
- type, 2-5
- WORD in displacement, 2-4

## Index

---

syntax statement in B 1000, translating to  
A Series syntax, 5-11

syntax-only compilation

example of, 1-3

of SORT programs, 1-3

SYSTEM/DUMPALL, use of, D-9

## T

tag sort

definition of, Glossary-8

in GSORT, E-6

tag-type, use of, 4-6

<tag-type>, 4-6

TAGCOBOL B 1000 statement, not  
supported by A Series, D-1

TAGRPG B 1000 statement, not supported  
by A Series, D-1

TAGSEARCH parameter statement, 4-9

<tagsearch statement>, 4-9

TAGSORT

B 1000 statement, conversion of, D-10

processing statement, 4-5

<tagsort statement>, 4-6

tape

designating number for

DISKANDTAPE processing statement,  
4-2

Integrated Tape and Disk sort, 4-2

requesting closure with release, D-5

TAPE in B 1000 FILE statement, conversion  
of, D-3

tape release

in FILE statement, 3-2

tape sort

default memory allocation, 4-7

designating number of, 4-7

designation of, 4-6

TAPESORT

A Series processing statement, 4-6

B 1000 statement, conversion of, D-11

<tapesort statement>, 4-7

task attribute

definition of, Glossary-8

TASKSTRING, using to submit CCRs, 5-2

TASKVALUE, meaning and testing of, 1-6

task variable, definition of, Glossary-8

TASKSTRING task attribute, accepting  
CCRs, 5-2

TASKVALUE task attribute, meaning and  
testing of, 1-6

TEACH B 1000 statement, (See  
REFORMAT compiler control  
option)

temporary compiler control record, use of,  
5-1

TEXT file type, compiling of, 1-3

TIME statement, conversion of, D-1

TIMING statement, conversion of, D-1

totals, in SORTS, E-7

TRAILING option, use in sign positioning,  
2-7

TRAILING SEPARATE option, use in sign  
positioning, 2-8

TRAILING ZONE option, use in sign  
positioning, 2-7

translating

B Series syntax to A Series syntax, 5-11  
messages, with MultiLingual System, 1-1

type, 2-5, (See also data type)

default units of, 2-3

designating in key fields, 3-5

list of valid literal types, 2-12

of file, defining, 1-2

SEQUENTIAL file, use of COMPILE  
command, 1-3

syntax of, 2-5

TEXT file, use of COMPILE command, 1-3

type of job, in Header record, E-11

type, of record, E-4

<type>, 2-5

types of key fields, E-33

types of printed messages, designating, E-13

types of sorted output files when GSORT  
option is set, E-6

## U

UA data type in B 1000 statements,  
conversion of, D-7

UPDATE, E-30

UDAY, E-30

UMONTH, E-30

UN data type in B 1000 statements,  
conversion of, D-7

unconditional forced field, use of, E-34

units

default type, 2-3

for MEMORY B 1000 statement, D-9

in key field, 3-4

unpacked data, specifying, E-26

unpacked decimal format, definition of  
 Glossary-9  
 unpacked decimal number, E-36  
 unsigned data, E-36  
 uppercase characters, use of, 1-1  
 user name, including in B 1000 FILE  
 statement, D-6  
 using Sort programs, 1-1  
 UYEAR, E-30

## V

valid literal types, table of, 2-12  
 value  
 assigning to sequence numbers, 5-12  
 compiler control record, use of, 5-1  
 compiler option, use of  
 Sequence Base, 5-12  
 Sequence Increment, 5-12  
 VERSION, 5-12  
 <value option>, 5-3  
 VARIABLE in B 1000 FILE statement,  
 conversion of, D-4  
 VERSION compiler control option, use of,  
 5-12  
 version information field  
 definition of, Glossary-9  
 method of setting, 5-12  
 <version option>, 5-13

## W

WFL, (See Work Flow Language)  
 word, definition of, Glossary-9  
 WORD, syntax of in displacement, 2-4  
 WORDS, as element in MEMORY parameter  
 statement, 4-8  
 work file  
 definition of, Glossary-9  
 description of, C-1  
 size designation, 4-10  
 Work Flow Language  
 comments, use of, 1-5  
 compiling Sort program, example of, 1-4  
 defaults for  
 ERRORLIMIT compiler control option,  
 5-5  
 GSORT compiler control option, 5-6  
 LIST compiler control option, 5-7

definition of, Glossary-9  
 immediate execution compilation, example  
 of, 1-5  
 initiating compilation from, 1-2  
 job disposition, list of, 1-4  
 Local Data Specification, use of, 1-4  
 SORT DATA CARD, use of, 1-4  
 using file attributes defined  
 at run time, 1-5  
 in compilation, 1-5  
 using to  
 compile Sort programs, 1-4  
 run Sort programs, 1-4  
 test TASKVALUE, 1-6  
 work tapes, designation of, 4-7  
 <work tapes>, 4-7  
 WORKFAMILY parameter statement, 4-10  
 <workfamily statement>, 4-10  
 WORKPACK B 1000 statement, conversion  
 of, D-11  
 WORKSIZE parameter statement, 4-10  
 <worksize statement>, 4-11  
 writing Sort program, 1-1

## Z

ZIP B 1000 statement, not supported by  
 A Series, D-1

## ZONE

data type  
 internal representation of, 2-8  
 use of, 2-7  
 data type in B 1000 statements, conversion  
 of, D-8  
 field associating with literal, 2-11  
 list of valid literal types, 2-12  
 use in sign positioning, 2-7  
 zone and digit comparison  
 specifying for alphanumeric data, E-25  
 specifying for numeric data, E-26  
 table of, F-1  
 zone only comparison  
 for binary values, E-26  
 specifying for alphanumeric data, E-25  
 specifying for numeric data, E-26  
 table of, F-3



## Index

---

- ?, (See question mark, use of in WFL)
- :, (See colon, use of to denote comments)
- ", (See quotation marks, use of in file designation)
- \$(See dollar sign, dollar sign)
- \$MERGE, use of, 2-2
- \$SEQUENCE, use of, 2-2
- %, (See percent sign, using to)
- #, (See number sign, purpose of)

# NOTES







116979400000380