60484800

2.3

CONTROL DATA CORPORATION

SORT/MERGE VERSION 5 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:

NOS 2

NOS/BE 1

SORT/MERGE PARAMETERS

PROCEDURE CALLS

60484800

CONTROL DATA CORPORATION

SORT/MERGE VERSION 5 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 2
NOS/BE 1

REVISION RECORD

Revision	Description
A (11/20/81)	Original release under NOS/BE 1 at PSR level 552.
B (02/26/82)	This manual is revised to reflect the support of NOS 2 at PSR level 552. Various technical and editorial changes have been made. This is a complete reprint.
C (02/20/84)	This manual is revised to reflect the support of NOS/BE 1.5 at PSR level 601. Various technical and editorial changes have been made.

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. Box 3492
SUNNYVALE, CALIFORNIA 94088-3492

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Inside Front Cover	C
Title Page	-
ii	C
iii/iv	C
v/vi	B
vii	C
viii	C
ix	A
1-1	B
1-2	B
1-3	A
1-4	B
2-1 thru 2-3	A
2-4	C
2-5	B
3-1 thru 3-11	C
4-1 thru 4-5	B
5-1	A
5-2 thru 5-7	C
6-1	A
7-1	A
7-2	A
7-3	C
7-4	C
8-1 thru 8-9	B
A-1 thru A-13	B
B-1	B
B-2	A
B-3	A
B-4 thru B-8	B
B-9	C
C-1 thru C-4	B
D-1	C
D-2	C
E-1 thru E-5	A
F-1	A
Index-1 thru -4	C
Comment Sheet/Mailer	C
Back Cover	-

PREFACE

This manual describes CONTROL DATA® Sort/Merge Version 5.0, a generalized sorting and merging utility. As described in this publication, Sort/Merge Version 5.0 operates under control of the NOS 2 and the NOS/BE 1 operating systems for the CDC® CYBER 180 and CYBER 170 Computer Systems.

This manual is written for the application programmer. You are assumed to be familiar with the CDC operating system installed at your site and, if you are using procedure calls to Sort/Merge, with the calling language.

You can find related information in the publications listed below; the publications are listed within groupings that indicate relative importance to you.

The NOS manual abstracts and the NOS/BE manual abstracts are pocket-sized manuals containing brief descriptions of the contents and intended audience of all NOS and NOS product set manuals and NOS/BE and NOS/BE product set manuals, respectively. The manual abstracts can be useful in determining which manuals are of greatest interest to you. The Software Publications Release History serves as a guide in determining which revision level of software documentation corresponds to the Programming System Report (PSR) level of software installed at your site.

The following manuals are of primary interest:

Publication	Publication Number	NOS 2	NOS/BE 1
COBOL Version 5 Reference Manual	60497100	X	X
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700	X	X
FORTRAN Version 5 Reference Manual	60481300	X	X
SORT/MERGE Version 5 Instant	60484900	X	X

The following manuals are of secondary interest:

Publication	Publication Number	NOS 2	NOS/BE 1
INTERCOM Version 5 Reference Manual	60455010		X
Network Products Remote Batch Facility Version 1 Reference Manual	60499600	X	
NOS Version 2 Manual Abstracts	60485500	X	
NOS Version 2 Reference Set, Volume 1 Introduction to Interactive Usage	60459660	X	
NOS Version 2 Reference Set, Volume 3 System Commands	60459680	X	
NOS/BE Version 1 Manual Abstracts	84000470		X
NOS/BE Version 1 Reference Manual	60493800		X
Software Publications Release History	60481000		X

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

FIGURES

TABLES

NOTATIONS

Unless otherwise specified, the conventions described in the following paragraphs are used throughout this manual in parameter and procedure call formats, examples, and diagnostic messages.

UPPERCASE	Uppercase letters indicate words, acronyms, or mnemonics required as input by Sort/Merge or produced as output by Sort/Merge. Parameter keywords and interactive commands are included in this category.
lowercase	Lowercase letters indicate words or symbols that you must supply.
...	Ellipsis in a parameter or procedure call format indicates that omitted portions of the format can be repeated.
[ ]	Brackets indicate an optional portion of a parameter or procedure call format.
..	Two periods in a parameter format indicate a range of letters or digits.
Δ	Delta indicates a space (blank).
$\boxed{1}\boxed{2}$	Boxes indicate character positions in storage.

Unless otherwise indicated, all numbers in text are decimal values.

GENERAL DESCRIPTION 1

Sort/Merge Version 5 is a set of powerful and efficient routines that provides high-speed sorting or merging of records with a single control statement. This section introduces the features of Sort/Merge and provides you with an overview of the Sort/Merge relationship with system software.

FUNCTIONS AND FEATURES

The purpose of sorting is to arrange items in order; the purpose of merging is to combine two or more sets of preordered items. Using Sort/Merge, records are rearranged in the order you specify, or two or more sorted files are combined into one file. Sort/Merge offers the following features:

- A sort or merge specification with a single control statement
- An optional directive file for sort or merge specification
- The capability of sorting or merging records from as many as 100 files with one call to Sort/Merge
- Character and noncharacter sort key types
- Five predefined collating sequences
- User-defined collating sequences
- Summing of numeric fields
- Sorting of any CYBER Record Manager (CRM) sequential file and any record type (except U)
- A tutorial dialog for interactive sort or merge specification
- A set of procedures for sort or merge processing from within a program written in FORTRAN or some other language
- Interface with COBOL programs through a COBOL5 control statement option or direct calls
- Owncode routines to insert, substitute, modify, or delete records during Sort/Merge processing

These features are described briefly in the following paragraphs.

Sorts are initiated with the SORT5 control statement; merges are initiated with the MERGE control statement. You specify processing requirements for the sort or merge with parameters in the respective control statement. Merge processing requires files that have already been sorted. Sort/Merge generates code based on the parameters and parameter values that you include in the SORT5 or MERGE control statement.

The SORT5 or the MERGE control statement can be up to 240 characters in length. You can specify more complex sorts or merges by using a directive file for additional parameters. Sort/Merge reads the parameters in a directive file after all of the parameters in the control statement have been read. Using a single control statement to specify your sort or merge provides simplicity and eliminates the necessity of creating, saving, and subsequently retrieving directive files. Directive files, on the other hand, provide standardization of installation parameters and collating sequences.

Records are sorted or merged on fields of data within each input record; the fields are called sort keys. Sort keys can be 6-bit character data, signed or unsigned binary integers, or floating-point numbers. Data can be sorted or merged according to a predefined or user-defined collating sequence for character data, or according to numeric value for numeric data. Numeric data can be signed or unsigned.

Records with equal sort key fields can be combined into one new record. You can specify that numeric fields be added to the corresponding fields in the other records with equal keys. The one resultant record contains the key fields and the numeric fields that are the sums of the specified numeric fields in all the records with equal keys; data fields that are not key or sum fields are set to the corresponding fields in one of the old records.

Sort/Merge uses CRM file processing and input/output capabilities. Any of the CRM sequential files or record types (except U) can be sorted or merged; this includes both fixed and variable length records. Because Sort/Merge uses CRM, the characteristics of files input to Sort/Merge and files output from Sort/Merge must be described to CRM with FILE control statements unless default characteristics apply.

You can specify any sort or merge from your terminal by entering the SORT5 or MERGE control statement and processing parameters. An interactive dialog is also available to guide you, step by step, through your sort or merge specification. The processing options are presented, and you choose the option you want as well as identify your files and sort key fields to Sort/Merge. Several interactive commands are available to control the use of the interactive dialog. During the dialog, you can obtain an explanation of Sort/Merge usage by entering HELP at your terminal.

A set of procedures exists that allows you to initiate a sort or merge from within your program written in FORTRAN or some other lanquage. The same processing options are available using calls to the procedures as are available using the SORT5 or the MERGE control statement; the procedure calls closely correspond to the Sort/Merge parameters.

A COBOL5 control statement option exists that allows you to use Sort/Merge 5. If you specify SORT5 in your COBOL5 control statement, Sort/Merge 5 is used when the COBOL SORT or MERGE statement in your program executes.

You can write routines, called owncode routines, to insert, substitute, modify, or delete records during Sort/Merge processing. Owncode routines are subroutines written in FORTRAN or some other language. The routines must be compiled; the file containing the compiled routines is loaded during your sort or merge. Depending on the exit number you specify, owncode routines are executed to process input records, output records, or records with equal keys. You can use owncode routines with control statement sorts or merges, or with procedure calls to Sort/Merge.

CONFIGURATION AND SOFTWARE RELATIONSHIPS

The hardware configuration needed to support Sort/Merge Version 5 is the same as the minimum configuration required for the NOS or NOS/BE operating system. One CDC CYBER 170 series host computer, together with peripheral processing units and disk subsystem is necessary. For installation, one tape unit is needed. Additionally, the interactive feature of Sort/Merge requires one 255x series communication subsystem.

Sort/Merge Version 5 resides in a library named SRT5LIB on the NOS or NOS/BE operating system. Sort/Merge requires only the basic NOS or NOS/BE software. Sort/ Merge interfaces with CYBER Record Manager, which is one of the standard products released as part of both operating systems.

Interactive use of Sort/Merge requires the Interactive Facility (IAF) Version 1 under NOS or INTERCOM Version 5 under NOS/BE. IAF in turn interfaces with the network software Network Access Method (NAM). The NOS Version 2 Reference Set, Volume 1 (for NOS users) or the INTERCOM reference manual (for NOS/BE users) contains information concerning terminal requirements.

JOB FLOW

You can use Sort/Merge for batch processing or interactively. For batch sort jobs, records to be sorted can be in permanent files, input in the same job with your SORT5 control statement and parameters, or supplied by an owncode routine. For merge processing, records must be in presorted files assigned to the job and named in the MERGE control statement. Sort/Merge uses CYBER Record Manager for input and output. Sorted or merged records can be written to a file and saved, listed by a printer, or processed by an owncode routine. Figure 1-1 illustrates batch job flow.

Interactively, records to be sorted or merged can be in local files or in permanent files you assign to your job; records to be sorted can also be supplied by an owncode routine. You can enter the SORT5 or MERGE control statement and processing parameters from your terminal, or you can use the interactive dialog to specify your sort or merge. At the end of the sort or merge, records are in a local file that can be displayed at your terminal, saved as a permanent file, or listed by a printer; output records can also be processed by an owncode routine. Figure 1-2 illustrates interactive job flow.

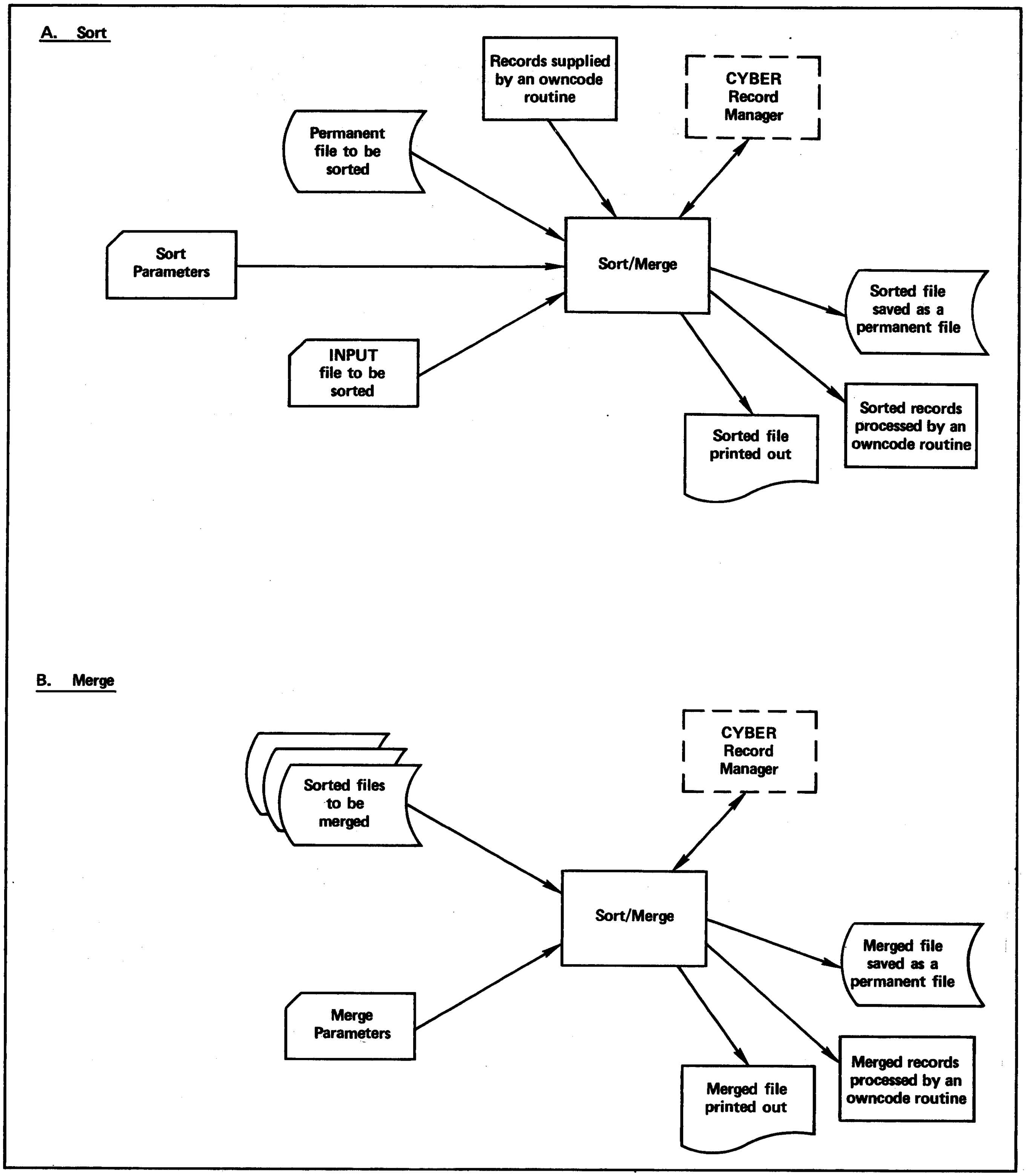

A. Sort
Records supplied by an owncode routine
CYBER Record Manager
Permanent file to be sorted
Sort Parameters
Sort/Merge
INPUT file to be sorted
Sorted file saved as a permanent file
Sorted records processed by an owncode routine
Sorted file printed out
B. Merge
CYBER Record Manager
Sorted files to be merged
Sort/Merge
Merge Parameters
Merged file saved as a permanent file
Merged records processed by an owncode routine
Merged file printed out

Figure 1-1. Batch Job Flow

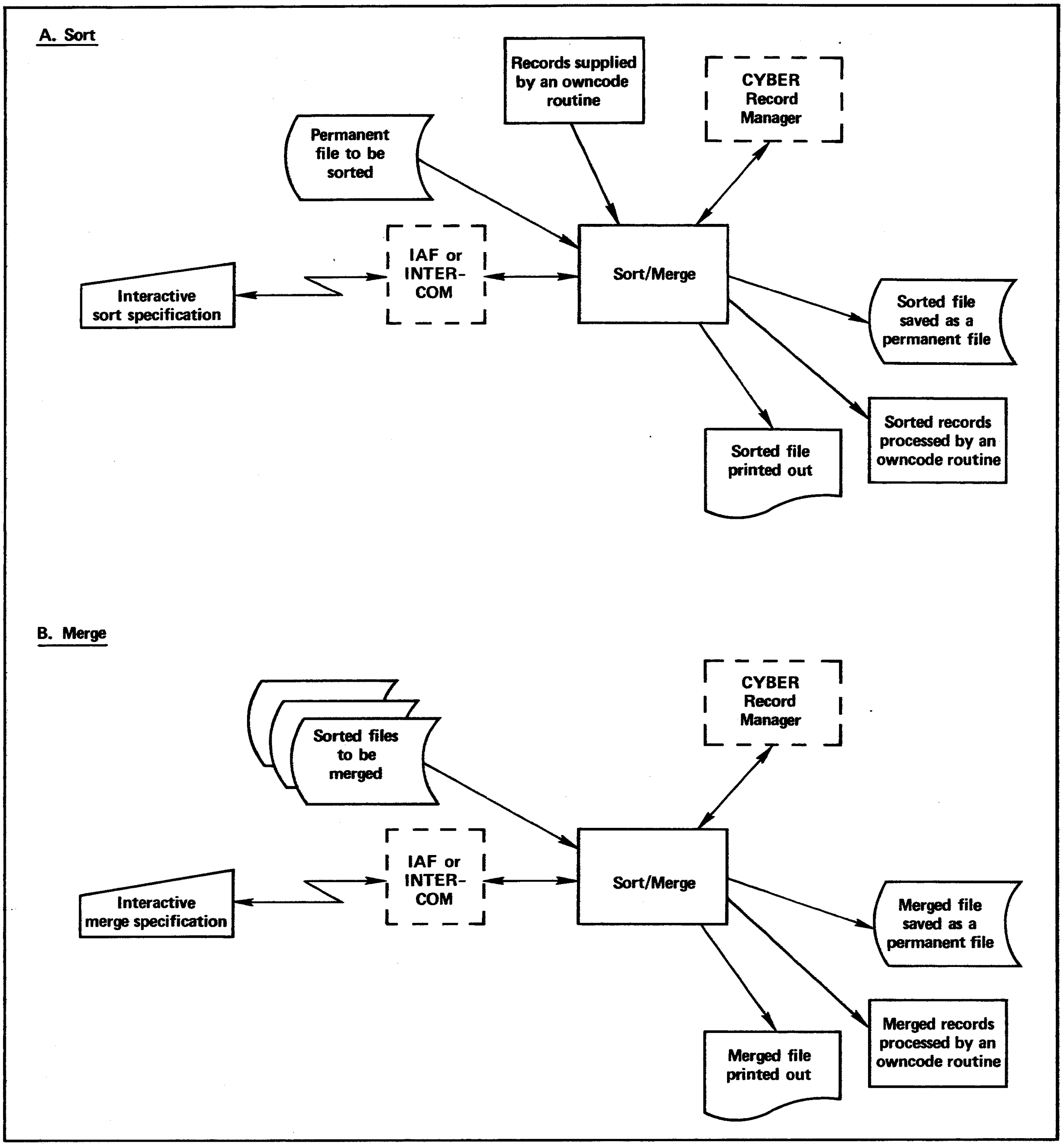

A. Sort
Records supplied by an owncode routine
CYBER Record Manager
Permanent file to be sorted
IAF or INTER-COM
Interactive sort specification
Sort/Merge
Sorted file saved as a permanent file
Sorted records processed by an owncode routine
Sorted file printed out
B. Merge
CYBER Record Manager
Sorted files to be merged
IAF or INTER-COM
Interactive merge specification
Sort/Merge
Merged file saved as a permanent file
Merged records processed by an owncode routine
Merged file printed out

Figure 1-2. Interactive Job Flow

SORT KEYS 2

All sort or merge operations are based on the ordering of values assigned to the data to be sorted or merged, which are called sort keys. This section includes an explanation of sort keys and a discussion of the information you must supply to describe sort keys to Sort/Merge.

DEFINITION

A sort key is a field of data within each record in an input file to be sorted or in a presorted file to be merged with one or more other files. The field is used by Sort/Merge to determine the order in which records are output. A file can be sorted or merged on more than one field of data; key fields must occur in the same position and be the same length in each record, and the total number of key characters must be less than 256 (if you also specify sum fields, the total number of characters in the key and sum fields together must be less than 256). Sort keys in files with variable length records cannot extend beyond the length of the shortest record in the file; the sort key field must be present in every record. For example, if the records range from a minimum of 25 characters to a maximum of 80 characters, all sort keys must be in the first 25 characters.

The first key you specify is the most important key and is called a major sort key. The keys you specify after the first key are of lesser importance and are called minor sort keys. Each sort key describes the length of the field, the position of the field within each record, the type of data expected in that field, and the effect of that data on the sort order. When you specify multiple sort keys, the keys can differ as to type of data, collating sequence, and sort order. The alignment of data in sort keys is important. Character data must be left-justified in the field; numbers must be right-justified in the field.

An example of a file with multiple sort keys is a university student file with a record for each student. Each record includes the last name and first and middle initials, the student number, the date of birth, the field of study, the grade point average, and a code representing class (freshman, sophomore, junior, senior); all the fields are written with character data. The file could be maintained with the student number as the major key - records are normally retrieved by specifying the student number. The file can be sorted by the name in standard alphabetic order when a list of student names is needed.

When a university department needs to know which students are majoring in fields within the department, the file can be sorted on the field of study. The same sort can specify the name as a minor key so that records with the same field of study are also sorted in alphabetic order by the name. The file can be sorted by the class code as the major key, and by the grade point average in descending numeric order as a minor key to give a list by class with the students having the highest grade point average at the beginning of the list.

DESCRIBING SORT KEYS

You must describe to Sort/Merge every field of data that you want used as a sort key. Sort key descriptions include the following information:

- Starting location of the key within the record
- Key length
- Type of data in the key field
- Sort order

You describe sort keys with either the KEY parameter in a control statement or directive file, or with the SM5KEY procedure call. The options and defaults for describing sort keys are discussed in the following paragraphs.

KEY LENGTH AND POSITION

You define key field length and position by specifying the first byte or bit of the field and either the number of bytes or bits in the field (length of the field), or the last byte or bit of the field. The leftmost byte and the leftmost bit in a record are counted as number 1. For character data, each character is six bits and occupies one byte. For example, if you want to specify the name of the university student file as a sort key, and the name field is the leftmost field in the record, you specify the first byte as 1. If the name field is 20 characters long, you specify length as 20.

Sort/Merge interprets the integers you specify for key length and position as bit numbers when the key type (discussed later in this section) specifies bits; otherwise, byte numbers are assumed. Key fields cannot overlap one another. No default exists for the starting location of a key within a record; you must specify the starting location. If you do not specify the length of a key, length defaults to 1.

KEY TYPE

You specify the type of data in a key field with the name of a collating sequence or with the name of a numeric data format. The data in a key field can be character or noncharacter. Character data is represented in the computer as display code values. To indicate the key type for character data, you specify the name of a collating sequence or, for numeric character data, the name of a numeric data format. Noncharacter data is represented in the computer as binary values. To indicate the key type for noncharacter data, you specify the name of a numeric data format.

Figure 2-1 illustrates the difference between the internal representation of character and noncharacter data. Table 2-1 summarizes character and noncharacter data types and the associated sort key type.

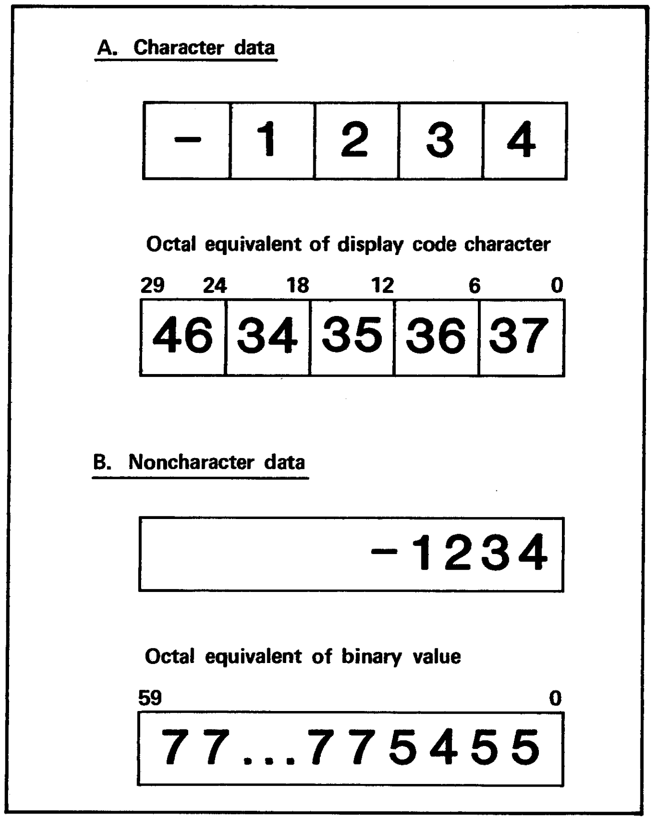


Figure 2-1. Internal Representation of Character and Noncharacter Data

If a sort key field contains any characters that are not meaningful for the key type you specify (an alphabetic character in a field defined as a numeric key for example), the sort order for that key field in that record is undefined. In the output file, the data for that key field in that record is also undefined. The record is still sorted according to any other sort keys you have specified.

The collating sequences and numeric data formats you can specify are discussed in the following paragraphs.

Collating Sequences

A collating sequence determines the precedence given to each character in relation to the other characters. You specify a collating sequence for character data to determine the sort order. The data must be display code characters.

Five predefined collating sequences are available to you as a Sort/Merge user: ASCII6, COBOL6, DISPLAY, EBCDIC6, and INTBCD. The default collating sequence is ASCII6. Table 2-2 shows the predefined collating sequences.

If none of the predefined collating sequences orders data as you would like, you can define a nonstandard collating sequence. If you are using control statement calls to Sort/Merge, you define your own collating sequence with SEQx parameters in the SORT5 or MERGE control statement. If you are using the procedure calls, you define your own collating sequence with SM5SEQx procedure calls.

Numeric Data Formats

Numeric data can appear in a key field in one of the formats listed in table 2-3. Numeric data can be signed or unsigned. For character numeric data that is signed, the sign can be a floating sign, an overpunch representation over the leading (leftmost) digit, a leading separate character, an overpunch representation over the trailing (rightmost) digit, or a trailing separate character.

A floating sign is a negative sign embedded between leading blanks and the numeric characters; leading zeros must be converted to blanks. Positive values in this format are not signed. The following examples are valid floating sign formats:

```
ΔΔ-1
ΔΔΔ1
ΔΔΔ0
1234
```

TABLE 2-1. DATA IN SORT KEY FIELDS

Type	Internal Representation	Data in Field	Type Specified by	Data Ordered According to
Character	Display code	Alphabetic	Name of a collating sequence	Specified collating sequence
		Numeric	Name of a collating sequence	Specified collating sequence
			Name of a numeric data format	Numeric value
Noncharacter	Binary value	Numeric	Name of a numeric data format	Numeric value

TABLE 2-2. PREDEFINED COLLATING SEQUENCES

Collating Sequence		ASCII6		COBOL6		DISPLAY		EBCDIC6		INTBCD	
Decimal	Octal	Graphics	Display Code	Graphics	Display Code	Graphics	Display Code	Graphics	Display Code	Graphics	CDC INTBCD Code
00	00	blank	55	blank	55	:†	00	blank	55	0	00
01	01	!	66	≤	74	A	01	.	57	1	01
02	02	"	64	%†	63†	B	02	<	72	2	02
03	03	#	60	[	61	C	03	(	51	3	03
04	04	$	53	→	65	D	04	+	45	4	04
05	05	%†	63†	≡	60	E	05	|	66	5	05
06	06	&	67	∧	67	F	06	&	67	6	06
07	07	'	70	↑	70	G	07	$	53	7	07
08	10	(	51	↓	71	H	10	*	47	8	10
09	11	)	52	>	73	I	11	)	52	9	11
10	12	*	47	≥	75	J	12	;	77	:	12
11	13	+	45	¬	76	K	13	¬	76	=	13
12	14	,	56	.	57	L	14	-	46	≠	14
13	15	-	46	)	52	M	15	/	50	≤	15
14	16	.	57	;	77	N	16	,	56	%	16
15	17	/	50	+	45	O	17	%†	63†	[	17
16	20	0	33	$	53	P	20	_	65	+	20
17	21	1	34	*	47	Q	21	>	73	A	21
18	22	2	35	-	46	R	22	?	71	B	22
19	23	3	36	/	50	S	23	:	00	C	23
20	24	4	37	,	56	T	24	#	60	D	24
21	25	5	40	(	51	U	25	@	74	E	25
22	26	6	41	=	54	V	26	'	70	F	26
23	27	7	42	≠	64	W	27	=	54	G	27
24	30	8	43	<	72	X	30	"	64	H	30
25	31	9	44	A	01	Y	31	¢	61	I	31
26	32	:†	00	B	02	Z	32	A	01	<	32
27	33	;	77	C	03	0	33	B	02	.	33
28	34	<	72	D	04	1	34	C	03	)	34
29	35	=	54	E	05	2	35	D	04	≥	35
30	36	>	73	F	06	3	36	E	05	¬	36
31	37	?	71	G	07	4	37	F	06	;	37
32	40	@	74	H	10	5	40	G	07	-	40
33	41	A	01	I	11	6	41	H	10	J	41
34	42	B	02	∨	66	7	42	I	11	K	42
35	43	C	03	J	12	8	43	!	62	L	43
36	44	D	04	K	13	9	44	J	12	M	44
37	45	E	05	L	14	+	45	K	13	N	45
38	46	F	06	M	15	-	46	L	14	O	46
39	47	G	07	N	16	*	47	M	15	P	47
40	50	H	10	O	17	/	50	N	16	Q	50
41	51	I	11	P	20	(	51	O	17	R	51
42	52	J	12	Q	21	)	52	P	20	∨	52
43	53	K	13	R	22	$	53	Q	21	$	53
44	54	L	14	]	62	=	54	R	22	*	54
45	55	M	15	S	23	blank	55	none	75	↑	55
46	56	N	16	T	24	,	56	S	23	↓	56
47	57	O	17	U	25	.	57	T	24	>	57
48	60	P	20	V	26	≡	60	U	25	blank	60
49	61	Q	21	W	27	[	61	V	26	/	61
50	62	R	22	X	30	]	62	W	27	S	62
51	63	S	23	Y	31	%†	63†	X	30	T	63
52	64	T	24	Z	32	≠	64	Y	31	U	64
53	65	U	25	:	00	→	65	Z	32	V	65
54	66	V	26	0	33	∨	66	0	33	W	66
55	67	W	27	1	34	∧	67	1	34	X	67
56	70	X	30	2	35	↑	70	2	35	Y	70
57	71	Y	31	3	36	↓	71	3	36	Z	71
58	72	Z	32	4	37	<	72	4	37	]	72
59	73	[	61	5	40	>	73	5	40	,	73
60	74	\	75	6	41	≤	74	6	41	(	74
61	75	]	62	7	42	≥	75	7	42	→	75
62	76	⌒	76	8	43	¬	76	8	43	≡	76
63	77	_	65	9	44	;	77	9	44	∧	77

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

TABLE 2-3. NUMERIC DATA FORMATS

Name	Data Type	Sign	Comments
NUMERIC_LO	Numeric characters	Leading overpunch	All characters are decimal digits except the leading character, which indicates a sign by an overpunch. Data is ordered according to numeric value with negative zero ordered before positive zero.
NUMERIC_LS	Numeric characters	Leading separate	All characters are decimal digits except the leading character, which is a negative or positive sign. Specifying a field that is not at least two characters in length causes a fatal error. Data is ordered according to numeric value with negative zero ordered before positive zero.
NUMERIC_TO	Numeric characters	Trailing overpunch	All characters are decimal digits except the trailing character, which indicates a sign by an overpunch. Data is ordered according to numeric value with negative zero ordered before positive zero.
NUMERIC_TS	Numeric characters	Trailing separate	All characters are decimal digits except the trailing character, which is a negative or positive sign. Specifying a field that is not at least two characters in length causes a fatal error. Data is ordered according to numeric value with negative zero ordered before positive zero.
NUMERIC_NS	Numeric characters	None	All characters are decimal digits. Data is ordered according to numeric value.
NUMERIC_FS	Leading blanks, numeric characters	- sign for negative values; a + character is not allowed	The field contains leading blanks (leading zeros must be converted to blanks); if the value is negative, the right-most leading blank must be converted to a minus sign. If the field contains no leading blanks, the value must be positive. This format is equivalent to the FORTRAN I format. Data is ordered according to numeric value with negative zero ordered before positive zero.
BINARY	Binary integer	None	The field starts and ends on character boundaries. Data is ordered according to numeric value.
BINARY_BITS	Binary integer	None	The field does not start or end on character boundaries. Data is ordered according to numeric value.
INTEGER	One's complement binary integer	Positive if leftmost bit is 0; negative if leftmost bit is 1	The field starts and ends on character boundaries. Data is ordered according to numeric value with negative zero ordered before positive zero.
INTEGER_BITS	One's complement binary integer	Positive if leftmost bit is 0; negative if leftmost bit is 1	The field does not start or end on character boundaries. Data is ordered according to numeric value with negative zero ordered before positive zero.
REAL	Normalized binary real or floating-point number	Signed	The field occupies a full computer word and is aligned on word boundaries. Data is ordered according to numeric value with all forms of zero ordered equally. The order of indefinite values is undefined.

The following examples are not valid floating sign formats:

Δ Δ 0 1	Leading zero not allowed
Δ - 0 1	Leading zero not allowed
+ 1 2 3	Positive sign not allowed
Δ Δ Δ Δ	All blank field not allowed

A negative sign overpunch is equivalent to overstriking a digit with a – , which is a punch in row 11. A positive sign overpunch is equivalent to overstriking a digit with a + , which is a punch in row 12. When a signed overpunch digit is received as input, the digit is punched as indicated in the second column of table 2-4. When a signed overpunch digit is displayed as output, the digit appears as indicated in the third column of table 2-4. Noncharacter numeric data can be signed or unsigned binary integers or normalized floating-point numbers.

You define numeric key fields by specifying the first byte of the field and either the length of the field or the last byte of the field. For types BINARY_BITS and INTEGER_BITS, you specify the first bit position of the field and either the length of the field in bits or the last bit of the field.

SORT ORDER

The order for sorting keys is ascending or descending. For numeric keys sorted in ascending order, the record having the key with the highest value is written last on the output file. Ascending character keys are sorted according to the collating sequence you specify. For numeric keys sorted in descending order, the record having the key with the lowest value is written last on the output file. Descending character keys are sorted in reverse order of the collating sequence you specify. The default sort order is ascending.

TABLE 2-4. SIGN OVERPUNCH REPRESENTATION

Sign and Digit	Input Punch	Output Representation
+9	12–9	I
+8	12–8	H
+7	12–7	G
+6	12–6	F
+5	12–5	E
+4	12–4	D
+3	12–3	C
+2	12–2	B
+1	12–1	A
+0	12–0	[†
–0	11–0	]†
–1	11–1	J
–2	11–2	K
–3	11–3	L
–4	11–4	M
–5	11–5	N
–6	11–6	O
–7	11–7	P
–8	11–8	Q
–9	11–9	R

†Valid on NOS with 029 keypunches. See appendix A, table A-2, for NOS/BE representations of +0 and –0.

CONTROL STATEMENT SORTS OR MERGES 3

This section describes Sort/Merge processing through control statement calls. The parameters that can appear in a SORT5 or in a MERGE control statement are discussed, along with optional directive files. The CYBER Record Manager (CRM) requirements for control statement sorts or merges are explained.

SORT5 CONTROL STATEMENT

Sort/Merge sort capabilities are available through the SORT5 control statement. Sort/Merge generates code based on the parameters you specify in the SORT5 control statement (and directive files, if used) or on default values. Sort/Merge generates the code to make efficient use of memory, mass storage, and the central processor.

The control statement begins with the word SORT5, which must be followed by a period. The control statement can appear in one of the following forms:

```
    SORT5.p1,p2,...,pn

    SORT5.p1Δp2Δ...Δpn
```

In the preceding control statement examples, p represents a processing parameter. Parameters must be separated by commas or spaces; commas and spaces can be used interchangeably.

The SORT5 control statement can be processed in a batch job or entered interactively from a terminal. The control statement must be CRM block type C (BT=C) and record type Z (RT=Z) format (see the CRM Basic Access Methods reference manual for information on block types and record types). Each line of the control statement can be up to 100 characters in length; however, characters in excess of 80 are ignored.

In interactive mode, control statements cannot be continued on more than one line. Instead, you can either create a directive file or use a procedure file so you can use a text editor to verify or change complex sort specifications.

In batch mode, you can continue a control statement on more than one line by terminating the line with two or more periods and beginning the new line with one period.

NOTE

Continuation is not allowed for ranges (indicated by two periods); ranges must be started and completed on the same line.

You can continue a control statement on any number of lines up to a maximum of 240 characters; periods used as continuation symbols are not included in the character count.

MERGE CONTROL STATEMENT

Sort/Merge merge capabilities are available through the MERGE control statement. The input files for a merge must be presorted. Input files for a merge with summing must be presorted and presummed. When called with the MERGE control statement, Sort/Merge generates code based on the parameters you specify in the control statement (and directive files, if used) or on default values. Sort/Merge generates code to make efficient use of memory, mass storage, and the central processor.

The control statement begins with the word MERGE, which must be followed by a period. The control statement can appear in one of the following forms:

```
    MERGE.p1,p2,...,pn

    MERGE.p1Δp2Δ...Δpn
```

In the preceding control statement examples, p represents a processing parameter. Parameters must be separated by commas or spaces; commas and spaces can be used interchangeably.

The MERGE control statement can be processed in a batch job or entered interactively from a terminal. The control statement must be CRM BT=C and RT=Z format. Each line of the control statement can be up to 100 characters in length; however, characters in excess of 80 are ignored.

In interactive mode, control statements cannot be continued on more than one line. Instead, you can either create a directive file or use a procedure file so you can use a text editor to verify or change complex sort specifications.

In batch mode, you can continue a control statement on more than one line by terminating the line with two or more periods and beginning the new line with one period. You can continue a control statement on any number of lines up to a maximum of 240 characters; periods used as continuation symbols are not included in the character count.

DIRECTIVE FILES

A control statement sort or merge can use optional directive files that contain processing parameters in addition to the parameters in the SORT5 or the MERGE control statement. Use of directive files allows specification of more complex sorts or merges (specifications that could be longer than allowed in a control statement) and permits standardization of installation parameters and collating sequences.

You can specify multiple directive files for a single control statement sort or merge. All directive files must be CRM sequential files of block type C (BT=C) and record type Z (RT=Z) with record length (FL) of 100 characters or less; however, characters in excess of 80 are ignored.

In a directive file, each statement can be up to 240 characters in length. Each statement must begin with the word SORT or with the word MERGE followed immediately by a comma or a space, and then followed by processing parameters, which are separated by commas or spaces. The following examples illustrate directive file formats:

```
SORT,p1,p2,...,pn

MERGE  p1Δp2Δ...Δpn
```

You can continue a directive file statement on more than one line by terminating the line with two or more periods. The new line must not begin with a period but can begin with one or more spaces. You specify directive files in the SORT5 or MERGE control statement with the DIR parameter, discussed in the Sort/Merge Parameters subsection.

SORT/MERGE PARAMETERS

You specify various parameters in the SORT5 or the MERGE control statement, or in a directive file, to direct Sort/Merge processing. Most parameters can appear either in a control statement or in a directive file. The DIALOG parameter cannot appear in a directive file. Unless stated otherwise, a parameter can appear only once in your sort or merge specification; a fatal error results if you specify duplicate parameters. You can specify parameters with keywords or by position in the control statement or directive file. When specified with keywords, parameters take one of the following forms, depending on the parameter:

```
keyword=((value-set)[,(value-set)]...)

keyword=value
```

Spaces can appear anywhere in a parameter except within a name, a number, or a range indicator. The order for specifying parameters by position follows the descriptions of the individual parameters. Refer to section 7 for information on owncode routines, which are mentioned in the descriptions of several of the parameters.

FROM

The FROM parameter (figure 3-1) specifies the input files from which the records to be sorted or merged are read. As many as 100 files can be sorted or merged in one job step. The files are read in the order that you specify them. The files are read past end-of-record marks; reading terminates when end-of-file or end-of-information is encountered (see the CRM Basic Access Methods reference manual). The files are rewound before and after use unless the name of the file is INPUT or a FILE control statement parameter specifies otherwise.

If you specify the file $NULL with the FROM parameter, records to be sorted are not read from a file; records must be supplied by an owncode routine. You can specify the FROM parameter more than once in a directive file. If you omit the FROM parameter, records to be sorted are read from the file OLD (unless an owncode routine supplies records). An empty FROM file results in a null sort.

```
FROM=lfn

FROM=(lfn1,lfn2,...)

FROM=$NULL

lfn        Logical file name; 1 to 7 letters or digits,
           beginning with a letter.
```

Figure 3-1. FROM Parameter Format

TO

The TO parameter (figure 3-2) specifies the file to which sorted or merged records are written (if records are left after owncode routine processing). The file is not rewound before or after use unless a FILE control statement parameter specifies otherwise.

```
TO=lfn

TO=$NULL

lfn        Logical file name; 1 to 7 letters or digits,
           beginning with a letter.
```

Figure 3-2. TO Parameter Format

If you specify the file $NULL with the TO parameter, sorted or merged records are not written to a file; you must supply an owncode routine to process all records. If you omit the TO parameter, records are written to the file NEW (unless an owncode routine processes all records).

KEY

The KEY parameter (figure 3-3) specifies the key fields that determine the sorted or merged order of output records. You can specify the KEY parameter more than once in a directive file on separate command lines. A maximum of 100 key fields can be specified.

Each key_def defines a separate key. They are defined in the order of precedence, from major key to minor key, read left to right. A key_def may consist of a range. A range consists of the first and last byte of the key, separated by two periods. A single integer may be considered as a special case of a range in which the first and last byte are the same. If a key_def consists of anything more than a range, it must be enclosed within parentheses, or else it will be processed as a multiple key sort.

The key position and length are the first byte of the field and either the number of bytes or bits in the field (length of the field) or the last byte or bit of the field. You specify first and length (or last) as integers; the leftmost byte and the leftmost bit in a record are counted as number 1. For character data, each character is six bits and occupies one byte. For all types of data, first

```
KEY = [(key_def [,key_def] ... )]
      [range                    ]

key_def   [range                          ]
          [(range[,type[,ad]])            ]
          [(first,length[,type[,ad]])     ]

range     [first       ]
          [first .. last]
```

first	First byte or bit of the key field.
last	Last byte or bit of the key field.
length	Number of bytes or bits in the key field; default is 1.
type	Name of a numeric data format or collating sequence; default is ASCII6.
ad	Order: A for ascending, D for descending; default is ascending.

Figure 3-3. KEY Parameter Format

and length (or last) refer to bytes except for the formats BINARY_BITS and INTEGER_BITS. You must specify the first byte or bit of a key field. If you do not specify otherwise in the KEY parameter, the key length is assumed to be 1.

The key type is the name of a predefined or user-defined collating sequence or the name of a numeric data format. If you do not specify a key type, the type is assumed to be the ASCII6 collating sequence. See section 2 for a list of predefined collating sequences and numeric data formats.

Sort order is either ascending or descending, and you specify A or D, respectively. If you specify neither, the sort order is assumed to be ascending. For numeric keys sorted in ascending order, the record having the key with the highest value is written last on the output file. Ascending character keys are sorted according to the collating sequence you specify. For numeric keys sorted in descending order, the record having the key with the lowest value is written last on the output file. Descending character keys are sorted in reverse order of the collating sequence you specify.

Records are first sorted according to the key field described by the leftmost key_def. Records with equal values for the major key field are sorted according to the field described by the next key_def, and so on. A fatal error occurs if key fields overlap one another. A fatal error also occurs if the total number of key characters is more than 255. (If you also specify sum fields, the total number of characters in the key and sum fields together must be less than 256.)

You must be careful when defining a sort with a single key. The following command defines a single key that is 20 bytes long, starting in byte 6:

```
SORT5. KEY=6..25
```

The following command defines the same sort key:

```
SORT5. KEY=((6,20))
```

The following command defines a sort on two single byte keys where byte 6 is the major key and byte 20 is the minor key:

```
SORT5. KEY=(6,20)
```

If you omit the KEY parameter, the following assumptions are made: KEY=1.. (minimum record length), the record length is the smallest MNR in the FILE control statements for input files (or the smallest FL for F and Z records), the key type is the ASCII6 collating sequence, and the sort order is ascending.

DIR

The DIR parameter (figure 3-4) specifies a directive file or files from which sort or merge parameters are read. Parameters in a directive file are read only after all of the control statement parameters have been read. The files are read in the order that you specify them. The files are rewound before and after use unless the name of the file is INPUT or a FILE control statement parameter specifies otherwise.

```
DIR=lfn

DIR=(lfn1,lfn2,...)
```

lfn	Logical file name; 1 to 7 letters or digits, beginning with a letter.

Figure 3-4. DIR Parameter Format

You can specify the DIR parameter more than once in a directive file. If you omit the DIR parameter, no parameters are read from a directive file; the sort or merge is completely specified in the SORT5 or the MERGE control statement.

L

The L parameter (figure 3-5) specifies the file to which listing information is written. The listing file is written in CRM BT=C, RT=Z format with a default full length of 132. Listing information includes the Sort/Merge version number, directive file statements, the time and date, and diagnostic messages, if any.

```
L=lfn

L=$NULL
```

lfn	Logical file name; 1 to 7 letters or digits, beginning with a letter.

Figure 3-5. L Parameter Format

The listing file is not rewound before or after use unless a FILE control statement parameter specifies rewind. If you specify the file $NULL with the L parameter, listing information is not written. If you omit the L parameter, listing information is written to the file OUTPUT.

E

The E parameter (figure 3-6) specifies the file to which diagnostic messages are written. The error file is written in CRM BT=C, RT=Z, and FL=72 format. The file is not rewound before or after use unless a FILE control statement parameter specifies to rewind.

```
E=lfn

E=$NULL

lfn       Logical file name; 1 to 7 letters or digits,
          beginning with a letter.
```

Figure 3-6. E Parameter Format

If you specify a listing file with a file name different from the error file name, diagnostic messages are written to the listing file and to the error file. If you specify the file $NULL with the E parameter, diagnostic messages are not written. If you omit the E parameter, diagnostic messages are written to the listing file.

EL

The EL parameter (figure 3-7) specifies the error level to be reported. If EL=T, all trivial errors, plus all errors of levels W, F, and C, are reported. If EL=W, all warning errors, plus all errors of levels F and C, are reported. If EL=F, all fatal errors, plus all errors of level C, are reported. If EL=C, all catastrophic errors are reported. Omitting the EL parameter is equivalent to EL=W.

```
EL=T

EL=W

EL=F

EL=C
```

Figure 3-7. EL Parameter Format

Errors are written to the error file or, if no error file is specified, to the listing file. See appendix B for an explanation of the four levels of errors.

DIALOG

The DIALOG parameter (figure 3-8) invokes an interactive dialog between you and Sort/Merge. The DIALOG parameter can appear only in a control statement. If DIALOG=YES, the dialog is invoked; if DIALOG=NO, the dialog is not invoked. Omitting the DIALOG parameter is equivalent to DIALOG=NO. The DIALOG parameter can be abbreviated as DIA, and Y or N can be specified instead of YES or NO. Any other parameters (except the STATUS parameter) specified in a control statement along with the DIALOG parameter do not affect sort or merge processing; the sort or merge is performed solely on the basis of your dialog responses. Interactive use of the dialog is described in section 4, and the text of the dialog is included in appendix E.

```
DIALOG=YES  (or DIA=Y)

DIALOG=NO   (or DIA=N)
```

Figure 3-8. DIALOG Parameter Format

ENR

The ENR parameter (figure 3-9) specifies the estimated number of records to be sorted or merged. You specify either a single value or a range of values, and decimal integers (from 0 through 1000000000) or the CYBER Control Language (CCL) variables R1, R2, R3, R1G, EF, or EFG. If ENR is specified as less than 1500, Sort/Merge uses less memory for the initial sort phase. This causes noticeably less memory required for the entire sort if the actual number of records is a low number.

```
ENR=expr

ENR=expr . . expr

expr      Expression that is a decimal integer, or CCL
          variable R1,R2,R3,R1G,EF, or EFG.
```

Figure 3-9. ENR Parameter Format

OWNF

The OWNF parameter (figure 3-10) specifies the file that is the source of owncode routines; the file, which must contain relocatable object modules, is loaded by Sort/Merge by means of the user call loader. The load map from loading the routines is written to file ZZZZZLM.

```
OWNF=lfn

lfn       Logical file name; 1 to 7 letters or digits,
          beginning with a letter.
```

Figure 3-10. OWNF Parameter Format

OWNFL

The OWNFL parameter (figure 3-11) specifies the exact number of characters in all records entering the sort from an owncode routine. The integer you specify can be from 1 to 5000 (larger records can be sorted if field length is increased). The OWNFL parameter can be abbreviated as OFL. See the discussion of the OWNMRL parameter for default values.

```
OWNFL=integer

OFL=integer
```

Figure 3-11. OWNFL Parameter Format

OWNMRL

The OWNMRL parameter (figure 3-12) specifies the maximum length in characters of any record entering the sort from an owncode routine. The integer you specify can be from 1 to 5000 (larger records can be sorted if field length is increased). The OWNMRL parameter can be abbreviated as OMRL. You do not need to specify the OWNMRL parameter if the sort has an input or output file with a maximum record length at least as long as the longest record supplied by an owncode routine.

```
OWNMRL=integer

OMRL=integer
```

Figure 3-12. OWNMRL Parameter Format

If you omit both the OWNMRL and the OWNFL parameters, the record length specification of records entering the sort from an owncode routine depends on the length specifications of the input and output files. If all input and output files have fixed length records of the same length, this length is used as a default OWNFL value; otherwise, the largest MRL or FL from any input or output file is used as a default OWNMRL value.

If the sort has no input or output files (records to be sorted are supplied by an owncode routine and sorted records are processed by an owncode routine), you must specify either OWNFL or OWNMRL; otherwise, a fatal error results. You cannot specify both the OWNFL and OWNMRL parameters for the same sort. Use of OWNFL rather than OWNMRL can increase performance.

OWNn

The OWNn parameter (figure 3-13) specifies the name of an owncode routine that is executed each time point n is reached during the sort or merge; n must be 1, 2, 3, 4, or 5. Owncode routines named by the OWNn parameter must be in the OWNF file. All owncode routines are loaded before the sort or merge is started. If you omit the OWNn parameter, no owncode routine is executed.

OWNn=proc

n	1, 2, 3, 4, or 5
proc	Procedure-name; 1 to 31 letters, digits or the special characters ? # @ _. Only 7 characters are significant and the first must be a letter.

Figure 3-13. OWNn Parameter Format

RETAIN

The RETAIN parameter (figure 3-14) directs Sort/Merge to output records with equal sort keys in the same order as the records are input (RETAIN=YES). If RETAIN=NO, or if you omit the RETAIN parameter, records with equal sort keys might not be output in the same order as they are input. If RETAIN=YES and you specify more than one sort or merge input file with the FROM parameter, the order in which you specify the files is the order in which records with equal keys are output. The RETAIN parameter can be abbreviated as RET, and Y or N can be specified instead of YES or NO.

```
RETAIN=YES   (or RET=Y)

RETAIN=NO    (or RET=N)
```

Figure 3-14. RETAIN Parameter Format

You cannot specify both the RETAIN and the SUM parameters in the same sort or merge. If you do specify both, a catastrophic error occurs.

SEQx

The SEQx parameters (figure 3-15) define your own collating sequence. The SEQN parameter signals the start of your collating sequence definition and names the sequence; name is used as the key type in the KEY parameter when records are sorted according to your collating sequence. Specifying a name that is the same as the name of one of the predefined collating sequences results in a fatal error.

```
SEQN=name
SEQS=('char', . . . , 'char')
.
.
.
[SEQR=YES or SEQR=Y]
[SEQA=YES or SEQA=Y]
```

name	Collating sequence name; 1 to 31 letters, digits, or the special characters ? # @ _. Only 10 characters are significant and the first must be a letter.
char	Character in the collating sequence.

Figure 3-15. SEQx Parameter Formats

Collating positions are assigned to the characters in your collating sequence by SEQS parameters; each SEQS parameter specifies either a single value step or a range of value steps in the sequence. A value step consists of one or more characters, and characters specified in the same value step collate equally. The first SEQS parameter specifies the first value step or range of steps, the second SEQS parameter specifies the second value step or range of steps, and so on until your collating sequence is completely defined. The following paragraphs describe how to specify steps in your collating sequence with SEQS parameters.

You can specify a single value step consisting of a single character by a graphic character (see appendix A) enclosed in apostrophes as in the following example:

```
SEQS='A'
```

You can also specify a single value step consisting of a single character by $CHAR(n) where n is the number of the character in the character set at your site (see table 3-1). The following example specifies a step consisting of the character number 08 in the character set, which is the letter H:

```
SEQS=$CHAR(08)
```

You can specify a single value step consisting of several characters by a single character enclosed in apostrophes, followed by one or more characters and/or ranges of characters. The characters are enclosed in apostrophes. All characters specified or implied in such a step collate equally. The following example specifies a step consisting of the blank and the digits 0, 1, 2, and 3, all of which collate equally:

```
SEQS=(' ','0','1','2','3')
```

The following example specifies the same value step as a range of characters where '0'..'3' specifies the range of digits from 0 through 3:

```
SEQS=(' ','0'..'3')
```

You can specify several value steps, each consisting of one character, with one value that is a range of characters. The characters are enclosed in apostrophes. The following example specifies a four-step collating sequence consisting of the digits 0, 1, 2, and 3, with 0 assigned the lowest collating value:

```
SEQS=('0'..'3')
```

TABLE 3-1. $CHAR(n) CHARACTER NUMBERS

Character Number (decimal)	CDC Graphic (64-Character Set)†	ASCII Graphic (64 Character Set)†	Octal 6-Bit Display Code	Character Number (decimal)	CDC Graphic (64-Character Set)†	ASCII Graphic (64 Character Set)†	Octal 6-Bit Display Code
00	:	:	00	32	5	5	40
01	A	A	01	33	6	6	41
02	B	B	02	34	7	7	42
03	C	C	03	35	8	8	43
04	D	D	04	36	9	9	44
05	E	E	05	37	+	+	45
06	F	F	06	38	-	-	46
07	G	G	07	39	*	*	47
08	H	H	10	40	/	/	50
09	I	I	11	41	(	(	51
10	J	J	12	42	)	)	52
11	K	K	13	43	$	$	53
12	L	L	14	44	=	=	54
13	M	M	15	45	blank	blank	55
14	N	N	16	46	,	,	56
15	O	O	17	47	.	.	57
16	P	P	20	48	≡	#	60
17	Q	Q	21	49	[	[	61
18	R	R	22	50	]	]	62
19	S	S	23	51	%	%	63
20	T	T	24	52	=	"	64
21	U	U	25	53	↱	_	65
22	V	V	26	54	∨	!	66
23	W	W	27	55	∧	&	67
24	X	X	30	56	↑	'	70
25	Y	Y	31	57	↓	?	71
26	Z	Z	32	58	<	<	72
27	0	0	33	59	>	>	73
28	1	1	34	60	≤	@	74
29	2	2	35	61	≥	\	75
30	3	3	36	62	¬	^	76
31	4	4	37	63	;	;	77

†For 63-character set differences, see the subsection entitled Character Set Anomalies in appendix A.

The preceding example is the same as specifying the following:

```
SEQS='0'
SEQS='1'
SEQS='2'
SEQS='3'
```

You can specify several value steps, each of which consists of more than one character, as ranges of characters where each range has the same number of characters. One value step in the collating sequence results for each character implied by the range. The characters are enclosed in apostrophes. A diagnostic message is issued if each range does not consist of the same number of characters. The following example specifies a three-step collating sequence where the first step consists of 1 and A (which are assigned the lowest collating value); the second step is 2 and B; the third step is 3 and C:

```
SEQS=('1'..'3','A'..'C')
```

The preceding example is the same as specifying the following:

```
SEQS=('1','A')
SEQS=('2','B')
SEQS=('3','C')
```

Any SEQS parameter that specifies a range of characters followed by a single character results in a fatal error. An apostrophe is specified in a SEQS parameter by four apostrophes (''''). You can specify the SEQS parameter more than once in a directive file. SEQS parameter use is summarized in table 3-2.

You can use one SEQR parameter to define a special value step that consists of all characters not explicitly or implicitly specified with SEQS parameters; either YES or Y is allowed. The following example defines a sequence in which the digits collate in numeric order, then all special characters (such as periods, commas, slashes) collate equally, and then letters collate in alphabetic order:

```
SEQS=('0'..'9')
SEQR=YES
SEQS=('A'..'Z')
```

You can specify one SEQA parameter to alter in output records characters in the same value step to the first character in the step; either YES or Y is allowed. The following example alters all asterisks and ampersands to slashes:

```
SEQS=('/','*','&')
SEQA=YES
```

Your collating sequence specification is terminated by any parameter other than SEQS, SEQR, or SEQA. You can define as many as 100 collating sequences by specifying a separate series of SEQx parameters for each collating sequence. The name of each user-defined collating sequence in a sort or merge must be unique. If you omit the SEQx parameters, a nonstandard collating sequence is not defined.

STATUS

The STATUS parameter (figure 3-16) specifies that a CYBER Control Language (CCL) variable be set to a value representing the highest level of error that occurred during the sort or merge. The CCL variables R1, R2, R3, R1G, EF, or EFG can be used for this purpose. Table 3-3 shows values that can be returned in the specified variable. For example, if you specify STATUS=R1 and R1 subsequently contains 20, a trivial error occurred.

STATUS=variable	**(or ST=variable)**
variable	**CCL variable R1, R2, R3, R1G, EF, or EFG.**

Figure 3-16. STATUS Parameter Format

If you include the STATUS parameter in your sort or merge specification, Sort/Merge does not abort if a catastrophic error occurs before any data records are input. However, if Sort/Merge calls another

TABLE 3-2. SEQS PARAMETER SUMMARY

Number of Value Steps Specified by One SEQS Parameter	Number of Characters in Each Value Step	How Specified	Example
1 step	1 character	Graphic character or $CHAR(n) where n is the number of the character in your character set.	SEQS='H' or SEQS=$CHAR(08)
1 step	More than one character	Each character in the step is enclosed in apostrophes and separated by commas, or a single character is followed by one or more ranges of characters.	SEQS=('I','O','X') or SEQS=(':','0'..'9')
More than one step	1 character	One range of characters.	SEQS=('0'..'9')
More than one step	More than one character	Ranges of characters where each range has the same number of characters.	SEQS=('A'..'E','V'..'Z')

TABLE 3-3. ERROR LEVEL CODES, STATUS PARAMETER

Error Level	Code
No errors	0
Trivial	20
Warning	30
Fatal	40
Catastrophic	50

product and an unrecoverable error results, an abnormal job termination will occur. For example, if Sort/Merge called a product, such as Common Memory Manager (CMM), and CMM processing results in an error, a termination will occur. If you omit the STATUS parameter, no error status variable is set, and Sort/Merge aborts if a catastrophic error occurs. The STATUS parameter can be abbreviated as ST.

SUM

The SUM parameter (figure 3-17) specifies the fields that are to be summed in records with equal key values. Input files for a merge with summing must be presorted and presummed. As many as 255 bytes can be summed. The records with all key fields equal are combined into one new record, and the other records with equal keys are deleted. The one resultant record contains the key fields and the fields that are the sums of the specified fields from all the records with equal keys; a data field that is not a key field or a sum field is set to the corresponding field from one of the old records.

```
SUM = ((sum_def)[,(sum_def)] . . . )

sum_def   [(first,length,[type[,rep]])]
          [(range[,type[,rep]])       ]

range     [first        ]
          [first .. last]

first     First byte or bit of the sum field.

last      Last byte or bit of the sum field.

length    Number of bytes or bits in the sum field;
          default is 1.

type      Name of a numeric data format (except
          REAL); default is INTEGER.

rep       Number of fields to be summed; default
          is 1.
```

Figure 3-17. SUM Parameter Format

All the fields to be summed must contain numeric data; otherwise, the contents of the new fields are undefined. The fields described as sum fields cannot also be key fields.

In figure 3-17, a key_def describes a field to be summed by specifying the field length and position, a numeric data format, and the number of fields to be summed. The field length and position are the first byte or bit of the field, and either the number of bytes or bits in the field (length of the field), or the last byte or bit of the field. You specify first and length (or last) as integers; the leftmost byte and the leftmost bit in a record are counted as number 1. First and length (or last) refer to bytes except for formats BINARY_BITS and INTEGER_BITS. You must specify the first byte or bit of the sum field. If you do not specify otherwise in the SUM parameter, the key length is assumed to be 1.

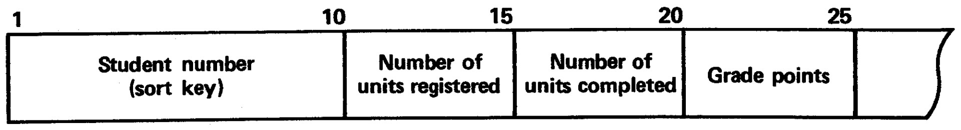
Sum field type is the name of a numeric data format except REAL; fields containing data in REAL format cannot be summed. You must specify the sum field type; no default exists. See section 2 for a list of numeric data formats. The number of fields to be summed is specified as an integer; the default is 1. If you specify multiple fields with one SUM parameter, the fields must be consecutive, must be the same length, and must contain the same type of numeric data. Sum fields cannot overlap one another. If a sum field contains no data because a record is short, the sum for that field is undefined. If you do not specify a sum type, the type is assumed to be INTEGER.

Figure 3-18 shows an example of a SUM parameter. Part A of figure 3-18 shows the layout of the record containing the fields to be summed. Part B shows three SUM parameters, each describing one field to be summed. Part C shows one equivalent SUM parameter that describes the three consecutive fields to be summed. Note that when you specify a single value-set, two sets of parentheses are still required.

If the summing of the specified fields results in new numeric values that do not fit into the sum fields, a fatal error results. One of the records that caused the overflow is deleted as usual, but the sum field contents in the remaining record are undefined. A diagnostic message indicating which field caused the overflow is issued, and the job step is terminated at the end of the sort (unless you include the STATUS parameter).

You can specify the SUM parameter more than once in a directive file. If you omit the SUM parameter, records with equal key values are not automatically combined into single records. You cannot specify both the SUM parameter and an owncode 5 routine for the same sort or merge. You cannot specify both the RETAIN parameter and the SUM parameter in the same sort or merge. If you do specify both, a catastrophic error occurs.

A. Record Layout


1
10
15
20
25
Student number (sort key)
Number of units registered
Number of units completed
Grade points

B. SUM Parameters

```
SUM=((11,5,INTEGER))
SUM=((16,5,INTEGER))
SUM=((21,5,INTEGER))
```

C. SUM Parameter

```
SUM=((11,5,INTEGER,3))
```

Figure 3-18. SUM Parameter Example

VERIFY

The VERIFY parameter (figure 3-19) directs Sort/Merge to check merge input records for correct order. This parameter applies only if you are performing a merge. If a merge input record is out of order when you have specified VERIFY=YES, the merge is terminated (regardless of the STATUS parameter) and a diagnostic message is issued. If VERIFY=NO, merge input records are not checked for correct order. Omitting the VERIFY parameter is equivalent to VERIFY=NO. You can specify VER instead of VERIFY, and Y or N instead of YES or NO. The VERIFY parameter is ignored if you are performing a sort.

```
VERIFY=YES     (or VER=Y)

VERIFY=NO      (or VER=N)
```

Figure 3-19. VERIFY Parameter Format

FASTIO

The FASTIO parameter (figure 3-20) specifies that certain sort or merge input and output records are to be read and written directly by Sort/Merge rather than by means of CYBER Record Manager (CRM). The FASTIO parameter is ignored during a merge. Files processed by the FASTIO parameter must reside on mass storage. The records in these files must be block type C and record type F, or block type I and record type W (see the CRM Basic Access Methods reference manual for detailed information). All input files must have identical record types and record lengths. If the file resides on tape, or if the records are not all the same type and length, a warning diagnostic is issued and the FASTIO option is canceled. The files are then processed using CRM. Sort/Merge obtains block and record type information from the file information table (FIT); a FILE control statement is not ignored. Specifying the FASTIO parameter results in increased speed, but not all errors are diagnosed; use this parameter when you are sure your files are correctly written and described.

```
FASTIO=YES     (or FASTIO=Y)

FASTIO=NO      (or FASTIO=N)
```

Figure 3-20. FASTIO Parameter Format

If FASTIO=YES, all sort or merge input and output files with block type C and record type F, or block type I and record type W, are read directly by Sort/Merge. If FASTIO=NO, all sort or merge input and output files are processed using CRM. Omitting the FASTIO parameter is equivalent to FASTIO=NO. You can specify Y or N instead of YES or NO.

SPECIFYING PARAMETERS BY POSITION

You can specify any of the Sort/Merge parameters in a control statement or in a directive file without a keyword by supplying a value in the position assigned to that particular keyword. Table 3-4 lists the parameter keywords and the position assigned to each. Omitted parameters are replaced by commas. For example, the following control statements are equivalent:

```
SORT5.INFL,OUTFL,5..9,,,,F

SORT5.FROM=INFL,TO=OUTFL,KEY=5..9,EL=F
```

The two preceding control statements specify that records to be sorted are in the file INFL, and sorted records are to be written to file OUTFL. The sort key field begins with the fifth character of each record, and the last character of the key field is the ninth character of the record. Fatal and catastrophic errors are to be reported.

You can mix keyword specification and positional specification of parameters in the same control statement or directive file. Whenever you specify a parameter with a keyword, the parameter retains its position number (table 3-4). Any positional parameters you specify after a keyword parameter are numbered according to the position number of

TABLE 3-4. PARAMETER POSITIONAL ORDER

Parameter Keyword	Position Number
FROM	1
TO	2
KEY	3
DIR	4
L	5
LO	6
E	7
EL	8
DIALOG	9
ENR	10
Reserved for future use	11
Reserved for future use	12
OWNF	13
OWNFL	14
OWNMRL	15
Reserved for future use	16
OWN1	17
OWN2	18
OWN3	19
OWN4	20
OWN5	21
RETAIN	22
SEQA	23
SEQN	24
SEQR	25
SEQS	26
STATUS	27
SUM	28
Reserved for future use	29
VERIFY	30
FASTIO	31

the last keyword parameter you specified. The following example illustrates mixing keyword and positional parameters:

```
SORT5.DIR=DIRFIL,,,,,,6000,RET=Y,,,,,EFG
```

The DIR parameter is position number 4. Commas replace parameters 5 through 9. The estimated number of records (6000) is position number 10. Numbering begins again with a keyword parameter (RETAIN number 22). Commas replace parameters 23 through 26. The positional value EFG is the status variable (STATUS number 27).

FILE CONTROL STATEMENT

Unless default characteristics apply for a file, you must specify the characteristics of all input and output data files named in a SORT5 or a MERGE control statement, or in a directive file, with a CYBER Record Manager (CRM) FILE control statement. FILE control statements must precede the SORT5 or the MERGE control statement. The format of the FILE control statement is shown in figure 3-21.

FILE(lfn[,keyword=option] . . .)

lfn	**Logical file name; 1 to 7 letters or digits, beginning with a letter.**
keyword=option	**Symbolic name of the file information table (FIT) field and the option selected.**

Figure 3-21. FILE Control Statement Format

The file information table (FIT) is a table through which file characteristics are communicated to CRM Basic Access Methods; the FIT contains information that describes the file and specifies how the file is accessed. You set FIT fields by specifying the symbolic name of the field in the FILE control statement and selecting one of the options associated with that field. Several FIT fields important to Sort/Merge processing are discussed in the following paragraphs. The CRM Basic Access Methods reference manual contains a complete list of all FIT fields and options that you can select. Note that specification of certain FIT fields can require setting of additional FIT fields.

Input files to be sorted or merged, and output files to which sorted or merged records are written, must be sequential files (records are read and written in order one after the other). The file organization (FO) FIT field can be included in the FILE control statement and sequential organization specified (FO=SQ); you can omit this parameter and sequential organization is assumed.

Sequential files are organized into groups of records called blocks. Block type is specified with the block type (BT) FIT field. The default is BT=C. CRM also offers several record types, and any of these record types except U can be sorted or merged. Record type is specified with the record type (RT) FIT field. The default is RT=Z. The full length (FL) or maximum record length (MRL) FIT field must be set to establish the number of characters in a record. The default is FL=150 or

MRL=150. The default is also a satisfactory description for records of less than 150 characters. If you have any files of considerably less than 150 characters (25 characters for example), Sort/Merge can perform more efficiently if you include a FILE control statement.

Default CRM characteristics apply for the files INPUT, OUTPUT, and PUNCH. You do not need to include FILE control statements for these files.

Table 3-5 summarizes the default file characteristics for Sort/Merge input and output data files, directive files, the error and listing files, and the files INPUT, OUTPUT, and PUNCH.

TABLE 3-5. FILE CHARACTERISTICS

FIT Field / File	BT	RT	FL
FROM files	C	Z	$FL \leq 150$
TO file	C	Z	$FL \leq 150$
DIR files	C	Z	$FL \leq 100$
L file	C	Z	FL=132
E file	C	Z	FL=72[†]
INPUT	C	Z	FL=80
OUTPUT	C	Z	FL=140
PUNCH	C	Z	FL=80

[†]If E=OUTPUT, FL=140 is used.

SPECIFYING MAXIMUM RECORD LENGTH

Unless default characteristics apply for a file, Sort/Merge requires that you specify maximum record length for input and output data files. You specify maximum record length by setting the FL or MRL FIT field in the FILE control statement. Set the FL FIT field for F or Z type records to specify the length of each record (if RT=F), or the upper limit of characters in a record (if RT=Z). For all other record types, set the MRL FIT field to specify the maximum length of a record.

FIELD LENGTH REQUIREMENTS

The minimum field length required for control statement sorts or merges is 55000_8 words; the default field length is 60000_8 words. Records up to 5000 characters in length can be sorted when your job is using the default field length. Large sort or merge operations can require additional central memory. The memory needed for a merge is directly proportional to the number of input files you have, because all files are simultaneously open. The default file buffer size (BFS) is 2001_8 words. You can change this value with the FILE control statement to save memory.

LISTINGS

If an error occurs during the sort or merge, the message SORT5 - USER ERROR, JOB ABORTED is sent to your job dayfile. Otherwise, when the sort or merge is complete, a message indicating the number of records sorted or merged is sent to the dayfile.

INTERACTIVE USE

Sort/Merge capabilities are available interactively; you can specify any of the Sort/Merge parameters from your terminal. Sort/Merge also provides a tutorial dialog to aid in specifying parameters. This section describes the use of this dialog, Sort/Merge interactive commands, and login and logout processes. The complete text of the dialog is included in appendix E.

In interactive mode, control statements cannot be continued on more than one line. Instead, you can either create a directive file or use a procedure file so you can use a text editor to verify or change complex sort specifications.

TERMINAL ACCESS TO SORT/MERGE

To access Sort/Merge from your terminal, you first establish the physical connection between the terminal and the computer. The method of establishing the connection varies depending on the type of terminal and the coupling between the terminal and computer. Consult the NOS Version 2 Reference Set, Volume 1 (for NOS users) or the INTERCOM reference manual (for NOS/BE users) for information on connection methods.

Connecting your terminal to the computer initiates the login process. The operating system, either NOS or NOS/BE, identifies itself to you, and you must supply information that indicates you are a valid system user.

LOGGING IN UNDER NOS

The login procedure to NOS begins with initial information from the system (including date, time, terminal name, installation name, and operating system version). The second line of this information is dependent on the installation. You are then prompted to enter information to indicate you are a valid system user. The prompts and required responses are as follows:

FAMILY:

> Enter the family name of the mass storage device that contains your permanent files. Some installations do not request a family name; the family name is the default family for the system.

USER NAME:

> Enter the user name assigned to you by your system administrator.

PASSWORD:

> Enter the 4- to 7-character password assigned to you to provide access security; some terminals overtype to preserve privacy.

termnam-APPLICATION:

> Enter IAF (for Interactive Facility, the network software that provides you with the time-sharing capabilities of NOS); termnam represents the terminal identification.

```
JSN: zzzz, NAMIAF
  CHARGE NUMBER:
```

or

```
JSN: zzzz, NAMIAF
 READY.
```

> If a charge number is requested, enter your assigned charge number and project number. The system will then respond with READY. The login procedure is now complete. You can then enter the batch subsystem. Do not change to ASCII mode since Sort/Merge does not process data in this mode.

Figure 4-1 shows an example of a NOS login. You are now ready to perform sort or merge operations as described later in this section.

```
82/01/08. 10.42.16. T143A
CDC NOS 2
FAMILY:
USER NAME: xxxxxxx
PASSWORD:  xxxx
T143A   - APPLICATION: iaf
JSN: AADI, NAMIAF
  CHARGE NUMBER:
? [redacted]
  PROJECT NUMBER:
? [redacted]

 READY.
/batch
RFL,0.
/
```

Figure 4-1. NOS Login Example

LOGGING IN UNDER NOS/BE

As soon as your terminal is connected to the system, the system responds with a message indicating date, time, and the INTERCOM version number (INTERCOM is the software that directs the flow of data between your terminal and the central site computer). The system requests the following:

PLEASE LOGIN

Enter LOGIN. You are then prompted for validation information. The prompts and required responses are as follows:

ENTER USER NAME-

> Enter the user name assigned to you by your system administrator.

XXXXXXXXXX ENTER PASSWORD-

> Enter the password (up to 10 letters and digits) assigned to you to provide access security; some terminals overtype to preserve privacy.

When the user number and password are accepted, a 2-character user code and the login time, followed by the equipment number and port number, are displayed or printed at your terminal. The message COMMAND- appears, indicating the login process is complete. Figure 4-2 shows an example of NOS/BE login. You are now ready to perform sort or merge operations as described in the following subsection.

INTERACTIVE DIALOG

Sort/Merge provides an interactive dialog to aid in specifying parameters. The tutorial dialog is designed to guide you, step by step, through your sort or merge specification. You can specify any control statement sort or merge, including a sort or merge with owncode routines and a nonstandard collating sequence, using the interactive dialog.

After login is complete, and before beginning the dialog, you must assign any files used in your sort or merge (files containing input records to be sorted, owncode routines, or presorted files to be merged) to your terminal with an ATTACH command or the equivalent. You must enter CYBER Record Manager (CRM) FILE control statements for any of your files that are not written in standard unit record format. Standard unit record format files include files created by punched cards in a batch job or lines entered at a terminal, and the special files INPUT, OUTPUT, and PUNCH. CRM defines these files as block type C (BT=C) and record type Z (RT=Z). Sort/Merge assumes each record contains 150 characters or less. If you have any files of considerably less than 150 characters (25 characters for example), Sort/Merge can perform more efficiently if you enter a FILE control statement. See your operating system reference manual for information about retrieving permanent files and the CRM Basic Access Methods reference manual for information about FILE control statements.

For NOS/BE users, Sort/Merge automatically connects and disconnects files INPUT and OUTPUT; Sort/Merge writes the dialog to OUTPUT, which is displayed or printed at your terminal. Your dialog responses are read from INPUT.

The interactive dialog begins when you type SORT5.DIALOG=YES or MERGE.DIALOG=YES. Any other parameters (except STATUS) that you enter along with DIALOG are ignored. During the dialog, Sort/Merge asks various questions, one at a time, and your response to each question determines the next question. For example, Sort/Merge asks if you have just one input file and no OWN1 or OWN2 subroutines. If you enter YES, the next question requests the name of your input file. If you enter NO, the next question asks if you have an OWN1 subroutine.

The expected response to each question is evident from the question. For example, one question asks how many key fields you have; you enter a number. For questions requiring yes or no answers, you can type YES or Y, or NO or N. If you make a mistake (misspell the word yes, for example), Sort/Merge gives you an explanation of the expected answer and asks the question again. If you enter an incorrect answer a second time, Sort/Merge asks you if you want help. If you type YES, an explanation of Sort/Merge usage is displayed or printed at your terminal. If you do not want help and type NO, the original question is repeated. Figure 4-3 shows an example of this sequence.

The last question Sort/Merge asks is how many records are to be sorted or merged. After you enter the appropriate number, Sort/Merge informs you that the sort or merge you have specified is beginning. When the sort or merge is complete, control of your terminal is returned to the operating system; the operating system prompt for input appears (/ under NOS, COMMAND- under NOS/BE).

```
CONTROL DATA INTERCOM 5.0
DATE  01/23/81
TIME  09.35.50.

PLEASE LOGIN
login
ENTER USER NAME-sort
[redacted] ENTER PASSWORD-

01/23/81  LOGGED IN AT  09.36.23.
          WITH USER-ID OK
          EQUIP/PORT 63/076


LOGIN      CREATED 01/21/81  TODAY IS 01/23/81

COMMAND-
```

Figure 4-2. NOS/BE Login Example

```
/sort5.dialog=yes <--------------------------------------- Begin interactive dialog.
 ARE YOU SURE THAT ALL YOUR FILES (IF ANY) ARE EITHER
 STANDARD UNIT RECORD FORMAT OR HAVE HAD FILE CONTROL
 STATEMENTS SPECIFIED FOR THEM
? yes
 DO YOU HAVE JUST ONE INPUT FILE AND NO OWN1 OR OWN2 SUBROUTINES
? no
 DO YOU HAVE AN OWN1 SUBROUTINE
? yse <-------------------------------------------------- Incorrect response.
 DO YOU HAVE AN OWN1 SUBROUTINE <------------------------ Question asked again.
? yse <-------------------------------------------------- Second incorrect response.
 DO YOU WANT HELP
? y
 SINCE YOU DID NOT SAY "NO" YOU ARE GOING TO BE HELPED.<- Help after second incorrect response.
 'Y' IS THE SAME AS 'YES' AND 'N' IS THE SAME AS 'NO'.
 TYPE 'RESTART' TO GO BACK TO THE BEGINNING.
 TYPE 'QUIT' TO STOP RUN.
 THE OWN1 ROUTINE CAN BE USED TO PROCESS INPUT RECORDS.  A DETAILED DESCRIPTION
 OF THIS OWNCODE ROUTINE IS IN THE SORT/MERGE REFERENCE MANUAL.
 DO YOU HAVE AN OWN1 SUBROUTINE <------------------------ Question asked again.
? yse
 DO YOU HAVE AN OWN1 SUBROUTINE
? yse
 DO YOU WANT HELP
? no
 DO YOU HAVE AN OWN1 SUBROUTINE <------------------------ No help after second incorrect
? yes                                                     response; question asked again.
```

Figure 4-3. Incorrect Response Example

If sorted or merged records are written to a file (not processed by an owncode routine), Sort/Merge creates a local file and gives this file the name you entered in response to the question "What is the name of your output file?" Upon completion of the sort or merge, you can display or print this file at your terminal, or route the file to a printer to be listed. If you want to use the file at a later time, you must save it as a permanent file before logging out; otherwise, the file is lost. Figure 4-4 shows an example of output file disposition.

INTERACTIVE COMMANDS

Several commands exist to control the use of the interactive dialog. These commands are HELP, RESTART, and QUIT. A command can be entered at any time during the dialog, but HELP, RESTART, and QUIT cannot be the names of user-supplied entities such as files and collating sequences. Use of the commands is described in the following paragraphs.

HELP

If you type HELP, an explanation of Sort/Merge usage is displayed or printed at your terminal. After the explanation, the question Sort/Merge asked you immediately before you entered the HELP command is asked again, resuming the dialog.

RESTART

If you type RESTART, the interactive dialog is started over at the beginning. All previous answers to questions are discarded. You can use this command if you give Sort/Merge incorrect information such as the wrong number of key fields.

QUIT

If you type QUIT, the interactive dialog is terminated. The sort or merge is not performed, and control of your terminal is returned to the operating system. You can then perform other terminal operations or logout. If permanent files containing input records to be sorted or merged, or owncode routines have been assigned to your job, you should release these files by using the RETURN command (see your operating system reference manual).

Figure 4-5 shows an example of interactive command use.

```
/sort5.dialog=yes <──────────────────────── Begin interactive dialog.
.
.
.
WHAT IS THE NAME OF YOUR OUTPUT FILE
? srtout <───────────────────────────────── Local file created for sorted output.
.
.
.
THANK YOU SORT/MERGE NOW BEGINS. <───────── Dialog completion; sort begins.
SORT5.dialog=YES <───────────────────────── NOS repeats control statement (up to 48 characters).
/copysbf,srtout,output <─────────────────── Display or print sorted file.
1REYES     S L 100246031558ANTHRO  3341 \
 MAYER     M I 100991122359ANTHRO  2882 |
 CHARLES   S H 101418032459ANTHRO  2453 |
 MARTIN    R C 100955082157ART     2891 |
 NEECE     M L  99911121358ART     2291 |
.                                        > <── File SRTOUT.
.                                       |
.                                       |
 LASEUR    P T 100678042256PSYCH   2233 |
 SUGARMAN  B T 100528070457SOC     3501 |
 SMITH     F R 101062120758SOC     2913 |
 DOUGLAS   M L 101325071558UNDEC   2585 |
 OKADA     N A 100103111750UNDEC   2225 /
 EOI ENCOUNTERED.
/rewind,srtout
REWIND,SRTOUT.
/copysbf,srtout,myfil <──────────────────── Copy sorted file to be listed by printer.
 EOI ENCOUNTERED.
/route,myfil,dc=pr <─────────────────────── Route copy of sorted file to printer.
 ROUTE COMPLETE.
/save,srtout <───────────────────────────── Make sorted file a permanent file.
```

Figure 4-4. Output File Disposition Example

```
/sort5.dialog=yes <─────────────────────────────────────────── Begin interactive dialog.
 ARE YOU SURE THAT ALL YOUR FILES (IF ANY) ARE EITHER
 STANDARD UNIT RECORD FORMAT OR HAVE HAD FILE CONTROL
 STATEMENTS SPECIFIED FOR THEM
? y
 DO YOU HAVE JUST ONE INPUT FILE AND NO OWN1 OR OWN2 SUBROUTINES
? n
 DO YOU HAVE AN OWN1 SUBROUTINE
? y
 WHAT IS THE NAME OF THE FILE CONTAINING YOUR OWNCODE SUBROUTINE(S)
? myrtns <───────────────────────────────────────────────────── Command entered.
 WHAT IS THE NAME OF YOUR OWN1 SUBROUTINE
? restart
 IN RESPONSE TO YOUR REQUEST RESTART HAS BEEN INITIATED.
 ARE YOU SURE THAT ALL YOUR FILES (IF ANY) ARE EITHER <──────── Dialog started over
 STANDARD UNIT RECORD FORMAT OR HAVE HAD FILE CONTROL           at beginning.
 STATEMENTS SPECIFIED FOR THEM
? y
 DO YOU HAVE JUST ONE INPUT FILE AND NO OWN1 OR OWN2 SUBROUTINES
? n
 DO YOU HAVE AN OWN1 SUBROUTINE
? y
 WHAT IS THE NAME OF THE FILE CONTAINING YOUR OWNCODE SUBROUTINE(S)
? ownsrtn
 WHAT IS THE NAME OF YOUR OWN1 SUBROUTINE
? myrtn
 DO YOU HAVE AN OWN2 SUBROUTINE
? help <─────────────────────────────────────────────────────── Command entered.
 'Y' IS THE SAME AS 'YES' AND 'N' IS THE SAME AS 'NO'.          \
 TYPE 'RESTART' TO GO BACK TO THE BEGINNING.                     | Usage
 TYPE 'QUIT' TO STOP RUN.                                        > explanation.
 THE OWN2 ROUTINE CAN BE USED TO PROCESS INPUT FILES.  A DETAILED DESCRIPTION
 OF THIS OWNCODE ROUTINE IS IN THE SORT/MERGE REFERENCE MANUAL.  /
 DO YOU HAVE AN OWN2 SUBROUTINE
? quit <─────────────────────────────────────────────────────── Command entered.
 SORT/MERGE IS QUITTING. <───────────────────────────────────── Sort/Merge terminated.
```

Figure 4-5. Interactive Command Example

SPECIFYING PARAMETERS INTERACTIVELY

You can specify any of the Sort/Merge parameters from your terminal in a SORT5 or MERGE control statement. After you login, assign all necessary files (including directive files) to your job, and type in FILE control statements for files with other than default characteristics. You enter SORT5 or MERGE followed by parameters in one of the following forms:

SORT5.p1,p2,...,pn

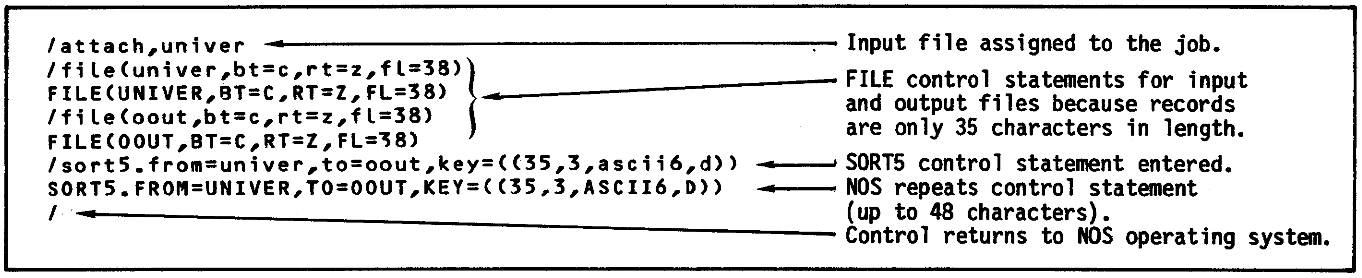
MERGE.p1Δ p2Δ ...Δ pn

In the preceding example, p represents a processing parameter. Parameters must be separated by commas or spaces; commas and spaces can be used interchangeably. You can enter the control statement and parameters in input line character positions 1 through 80. If your sort or merge specification is longer than 80 characters, you must use a directive file for the additional parameters.

Figure 4-6 shows an example of specifying parameters interactively.

LOGGING OUT

To end your terminal session, you type GOODBYE, BYE, or LOGOUT if you are using the Interactive Facility of NOS. If you are using INTERCOM under NOS/BE, you type LOGOUT. Under either system, a message indicating that you are logged out is displayed or printed at your terminal.



```
/attach,univer                                          <---> Input file assigned to the job.
/file(univer,bt=c,rt=z,fl=38) }                              FILE control statements for input
FILE(UNIVER,BT=C,RT=Z,FL=38)  }<---------------------------- and output files because records
/file(oout,bt=c,rt=z,fl=38)   }                              are only 35 characters in length.
FILE(OOUT,BT=C,RT=Z,FL=38)    }
/sort5.from=univer,to=oout,key=((35,3,ascii6,d))      <----> SORT5 control statement entered.
SORT5.FROM=UNIVER,TO=OOUT,KEY=((35,3,ASCII6,D))       <----> NOS repeats control statement
/ <-------------------------------------------------------->  (up to 48 characters).
                                                             Control returns to NOS operating system.
```

Figure 4-6. Interactive Parameter Specification Example

PROCEDURE CALLS 5

Sort/Merge also exists as a set of procedures; a sort or merge can be invoked from within a program by a sequence of procedure calls. The procedures can be called from FORTRAN or by a program in any language that uses the standard FORTRAN calling sequence. The same type of Sort/Merge processing is available through the procedures as is available through the SORT5 or the MERGE control statement; the procedure calls closely correspond to the Sort/Merge parameters. This section describes the procedure calls; COBOL programmers should see section 6 for information on calling the procedures.

All conventions for coding FORTRAN statements must be observed in procedure calls (see the FORTRAN 5 reference manual). You must assign any permanent files named in procedure calls to your job with an ATTACH command (or the equivalent) in the control statement section of the job. The procedures cannot be used in a static memory management environment. A merge could require twice as much mass storage for intermediate merge files as is required for the merge input files.

All files to be sorted, or presorted files to be merged, must be sequential files. The default file characteristics are block type C, record type Z, and full length of 150 characters or less. For files with other block types and record types, you must include CYBER Record Manager (CRM) FILE control statements in the control statement section of your job. See the CRM Basic Access Methods reference manual for information on block types and record types.

The procedures can be called in any order with two exceptions: SM5SORT or SM5MERG must be the first procedure called, and SM5END must be the last procedure called. Sort/Merge collects processing information until SM5END is called; the sort or merge is then performed. Unless stated otherwise, a procedure can be called only once during a sort or merge. Refer to section 7 for information on owncode routines, which are mentioned in the descriptions of several of the procedures.

SM5SORT

Execution of the SM5SORT procedure signals the beginning of a sort specification. The SM5SORT procedure must be the first procedure called for a sort. The format of the SM5SORT call is shown in figure 5-1.

```
CALL SM5SORT(option)

option      Result array name (1 to 7 letters or digits,
            beginning with a letter) or the integer 0.
```

Figure 5-1. SM5SORT Format

The array associated with the SM5SORT procedure is a 16-element integer array that returns sort statistics and results to your program when the sort is completed. You set the first element of the result array to the number of elements (as many as 15) in which you want to receive information. If you set the first element to 0 or specify CALL SM5SORT(0), no statistics or results are returned. The type of result that is returned in each element is shown in table 5-1.

TABLE 5-1. RESULT ARRAY FORMAT

Array Element Number	Contents
1	Number of elements of results you want returned (0 through 15)
2	Number of records read from sort or merge input files
3	Number of records deleted by an owncode 1 routine
4	Number of records inserted by an owncode 1 routine
5	Number of records inserted by an owncode 2 routine
6	Number of records sorted or merged
7	Number of records deleted by an owncode 3 routine
8	Number of records inserted by an owncode 3 routine
9	Number of records inserted by an owncode 4 routine
10	Reserved
11	Number of records deleted by an owncode 5 routine
12	Number of records combined by summing
13	Number of records written to output file
14	Minimum record length in characters
15	Average record length in characters
16	Maximum record length in characters

SM5MERG

Execution of the SM5MERG procedure signals the beginning of a merge specification. The SM5MERG procedure must be the first procedure called for a merge. The format of the SM5MERG call is shown in figure 5-2.

CALL SM5MERG(option)

option	Result array name (1 to 7 letters or digits, beginning with a letter) or the integer 0.

Figure 5-2. SM5MERG Format

The input files for a merge must be presorted. Input files for a merge with summing must be presorted and presummed.

The array asociated with the SM5MERG procedure is a 16-element integer array that returns merge statistics and results to your program when the merge is completed. You set the first element of the result array to the number of elements (as many as 15) in which you want to receive information. If you set the first element to 0 or specify CALL SM5MERG(0), no statistics or results are returned. The type of result that is returned in each element is shown in table 5-1. Note that the array is not set by Sort/Merge until the end of the merge.

SM5FROM

Execution of the SM5FROM procedure specifies the input files from which the records to be sorted or merged are read. The format of the SM5FROM call is shown in figure 5-3.

CALL SM5FROM('file1'[,'file2'] . . .)

'file'	File name; 1 to 7 letters or digits, beginning with a letter.

Figure 5-3. SM5FROM Format

You can specify as many as 100 files with one SM5FROM procedure, or you can call the SM5FROM procedure as many as 100 times. The files are read in the order that you specify them. The files are rewound before and after use unless the name of the file is INPUT or a FILE control statement parameter specifies otherwise.

Sort/Merge assumes that the files you specify with the SM5FROM procedure are CRM block type C and record type Z with FL of 150 characters or less. If you specify files that have different characteristics, you must describe these files to Sort/Merge by including FILE control statements in the control statement section of your job. The files are not read past embedded end-of-record or end-of-file marks (see the CRM Basic Access Methods reference manual).

If the SM5FROM procedure is not called, records to be sorted are read from the file OLD (unless an owncode routine supplies records). An empty SM5FROM file results in a null sort.

SM5TO

Execution of the SM5TO procedure specifies the file to which sorted or merged records are written (if records are left after owncode routine processing). The format of the SM5TO call is shown in figure 5-4.

CALL SM5TO('file')

'file'	File name; 1 to 7 letters or digits, beginning with a letter.

Figure 5-4. SM5TO Format

The file is not rewound before or after use unless the name of the file is OUTPUT or a FILE control statement parameter specifies otherwise. Sort/Merge assumes that the file you specify with the SM5TO procedure is CRM block type C and record type Z with FL of 150 characters or less. If you specify a file with different characteristics, you must describe the file to Sort/Merge by including a FILE control statement in the control statement section of your job.

Sort/Merge writes the file directly, and the file is closed when the sort or merge is completed unless the name of the file is INPUT. If your program also reads or writes the file, you must include an OPEN or REWIND statement in your program after calling the procedures. If the SM5TO procedure is not called, records are written to the file NEW (unless an owncode routine processes all records).

SM5KEY

Execution of the SM5KEY procedure specifies a single key field for the sort or merge. The format of the SM5KEY call is shown in figure 5-5.

CALL SM5KEY(first,length,'type','ad')

first	First byte or bit of the key field.
length	Number of bytes or bits in the key field.
'type'	Name of a numeric data format or collating sequence.
'ad'	Order, either 'A' for ascending or 'D' for descending.

Figure 5-5. SM5KEY Format

In the SM5KEY procedure call, you specify the first byte or bit of the key field, the length of the field in bytes or bits, the key type (the name of a collating sequence or a numeric data format), and the sort order. You must specify the key characteristics in the order shown.

You specify first and length as integers; the leftmost byte or bit in a record is counted as number 1. For character data, each character is six bits and occupies one byte. For numeric data, first and length refer to bytes except for formats BINARY_BITS and INTEGER_BITS. You must specify the first byte or bit of a key field. If you do not specify the length, length defaults to 1.

The key type is the name of a predefined collating sequence or your own collating sequence defined by SM5SEQx procedures, or the name of a numeric data format. The name must be enclosed in apostrophes. The default type is the ASCII6 collating sequence. See section 2 for a list of predefined collating sequences and numeric data formats.

The sort order is either ascending or descending, and you specify 'A' or 'D', respectively. The default order is ascending. For numeric keys sorted in ascending order, the record having the key with the highest value is written last on the output file. Ascending character keys are sorted according to the collating sequence you specify. For numeric keys sorted in descending order, the record having the key with the lowest value is written last on the output file. Descending character keys are sorted in reverse order of the collating sequence you specify.

You can call the SM5KEY procedure as many as 100 times during a sort or merge to specify multiple sort keys. Output records are sorted or merged according to the key field described by the first SM5KEY procedure called, then according to the key field described by the second SM5KEY procedure called, and so on. The total number of key characters must be less than 256 (if you also specify sum fields, the total number of characters in the key and sum fields together must be less than 256). Key fields cannot overlap one another. If the SM5KEY procedure is not called, the entire record is used as a character key field, and the default key type and sort order are used.

SM5E

Execution of the SM5E procedure specifies the file to which diagnostic messages are written. The format of the SM5E call is shown in figure 5-6.

CALL SM5E('file')

'file'	File name; 1 to 7 letters or digits, beginning with a letter.

Figure 5-6. SM5E Format

The file is written in CRM BT=C, RT=Z, and FL=72 format. The file is not rewound before or after use unless a FILE control statement parameter specifies rewind. If the SM5E procedure is not called, diagnostic messages are written to the file OUTPUT. If you specify the file $NULL with the SM5E procedure, diagnostic messages are not written.

SM5EL

Execution of the SM5EL procedure specifies the error level to be reported. The format of the SM5EL call is shown in figure 5-7.

CALL SM5EL(error-level)

error-level	Alphabetic character enclosed in apostrophes or integer (see table 5-2).

Figure 5-7. SM5EL Format

The error levels that you can select are shown in table 5-2. You can specify the alphabetic character enclosed in apostrophes or the integer. For example if you specify 'W' or 2, any warning, fatal, and catastrophic error messages are reported. Errors are written to the file specified by the SM5E procedure or, if the SM5E procedure is not called, to the file OUTPUT. If the SM5EL procedure is not called, errors of levels T, W, F, and C are reported. See appendix B for an explanation of the four levels of errors.

TABLE 5-2. ERROR LEVEL SPECIFICATION

Error Level	Errors Reported
'T' or 1	T, W, F, C
'W' or 2	W, F, C
'F' or 3	F, C
'C' or 4	C

SM5ENR

Execution of the SM5ENR procedure specifies the estimated number of records to be sorted or merged. The format of the SM5ENR call is shown in figure 5-8.

CALL SM5ENR(integer)

integer	Number of records to be sorted or merged; 0 to 1000000000.

Figure 5-8. SM5ENR Format

The integer you specify is the approximate number of records to be sorted or merged. If SM5ENR is specified as less than 1500, Sort/Merge uses less memory for the initial sort phase. This causes noticeably less memory required for the entire sort if the actual number of records is a low number.

SM5FAST

Execution of the SM5FAST procedure can specify that certain sort input and output records are to be read and written directly by Sort/Merge rather than by means of CYBER Record Manager (CRM). The SM5FAST procedure is ignored during a merge. The format of the SM5FAST call is shown in figure 5-9.

```
CALL SM5FAST(option)

option        'YES'  or  'Y'
              'NO'   or  'N'
```

Figure 5-9. SM5FAST Format

The option can be 'YES' or 'NO' (or 'Y' or 'N'). If the option is 'YES', certain sort or merge input and output records are read and written directly by Sort/Merge. If the option is 'NO', CRM is used.

Files processed by the SM5FAST procedure must reside on mass storage. The records in these files must be block type C and record type F, or block type I and record type W (see the CRM Basic Access Methods reference manual for detailed information). All input files must have an identical record type and record length. If the file resides on tape, or if the records are not all the same type and length, a warning diagnostic is issued and the SM5FAST option is canceled. The files are then processed using CRM. Sort/Merge obtains block and record type information from the file information table (FIT); a FILE control statement is not ignored. Calling the SM5FAST procedure with 'YES' results in increased speed, but not all errors are diagnosed; use this proce- dure when you are sure your files are correctly written and described.

SM5OFL

Execution of the SM5OFL procedure specifies the number of characters in fixed length records entering the sort from an owncode routine. The format of the SM5OFL call is shown in figure 5-10.

```
CALL SM5OFL(integer)
```

Figure 5-10. SM5OFL Format

The integer you specify is the exact number of characters in each record; a catastrophic error results if a record entering the sort from an owncode routine does not have the exact number of characters. If the SM5OFL procedure is not called, records entering the sort from an owncode routine can be no longer than the longest allowed input or output record. You cannot call both the SM5OFL procedure and the SM5OMRL procedure in the same sort.

SM5OMRL

Execution of the SM5OMRL procedure specifies the maximum length of any record entering the sort from an owncode routine. The format of the SM5OMRL call is shown in figure 5-11.

```
CALL SM5OMRL(integer)
```

Figure 5-11. SM5OMRL Format

The integer you specify is the maximum record length in characters. (The SM5OFL procedure is recommended if all records entering the sort from an owncode routine are the same length.) The SM5OMRL procedure need not be called if the sort has an input or output file with a maximum record length at least as long as the record length specified by this procedure. If the SM5OMRL procedure is not called, records entering the sort from an owncode routine can be no longer than the longest allowed input or output record. You cannot call both the SM5OMRL procedure and the SM5OFL procedure in the same sort.

SM5OWNn

Execution of the SM5OWNn procedure specifies an owncode routine that is executed each time a certain point is reached during the sort or merge. The format of the SM5OWNn call is shown in figure 5-12.

```
CALL SM5OWNn(proc)

n        1, 2, 3, 4, or 5

proc     Owncode routine name; 1 to 7 letters or digits,
         beginning with a letter.
```

Figure 5-12. SM5OWNn Format

The procedure you specify is the name (address) of the owncode routine; in a FORTRAN program, the name must be declared in an EXTERNAL statement. The n associated with the SM5OWNn call must be 1, 2, 3, 4, or 5. If the SM5OWNn procedure is not called, no owncode routine is executed.

SM5RETA

Execution of the SM5RETA procedure can specify that records with equal sort keys are output in the same order as they are input. The format of the SM5RETA call is shown in figure 5-13.

```
CALL SM5RETA(option)

option        'YES'  or  'Y'
              'NO'   or  'N'
```

Figure 5-13. SM5RETA Format

The option can be 'YES' or 'NO' (or 'Y' or 'N'). If the option is 'YES', records with equal keys retain their original order. If the option is 'NO', records with equal keys might not retain their original order.

If RETAIN=YES and you specify more than one sort or merge input file with the SM5FROM procedure, the order in which you specify the input files is the order in which records with equal keys are output. If the SM5RETA procedure is not called, records with equal keys might not be output in the same order as they are input.

You cannot specify both the SM5RETA parameter and the SM5SUM parameter in the same sort or merge. If you do specify both, a catastrophic error occurs.

SM5SEQx

Execution of a series of SM5SEQx procedures specifies a user-defined collating sequence that is referenced by the SM5KEY procedure. The formats of the SM5SEQx calls are shown in figure 5-14.

```
CALL SM5SEQN('name')
CALL SM5SEQS('char' , . . . , 'char')
.
.
.
[CALL SM5SEQR('YES') or CALL SM5SEQR('Y')
[CALL SM5SEQA('YES') or CALL SM5SEQA('Y')
```

'name'	Collating sequence name; 1 to 31 letters, digits, or the special characters ? # @ –. Only 10 characters are significant and the first must be a letter.
'char'	Character in the collating sequence.

Figure 5-14. SM5SEQx Formats

The SM5SEQN procedure names your collating sequence. The name is a literal enclosed in apostrophes; the name cannot be the same as the name of one of the predefined collating sequences. The SM5SEQN procedure must be the first called in the series of SM5SEQx procedures.

Collating positions are assigned to the characters in your collating sequence by the SM5SEQS procedure, which can be called more than once. Each SM5SEQS procedure specifies a value step in the sequence. A value step consists of one or more characters, and characters specified in the same step collate equally. Characters specified with the first SM5SEQS procedure called collate first, characters specified with the second SM5SEQS procedure called collate second, and so on. The following paragraphs describe how to specify steps in your collating sequence with SM5SEQS procedures.

You can specify a value step consisting of one character by a graphic character (see appendix A) enclosed in apostrophes as in the following example:

```
    CALL SM5SEQS('A')
```

You can specify a value step consisting of several characters by single characters enclosed in apostrophes and separated by commas. The following example specifies a step consisting of the blank and the digits 0, 1, 2, and 3, all of which collate equally:

```
    CALL SM5SEQS(' ','0','1','2','3')
```

If the SM5SEQR procedure is called, equal collating position is assigned to any characters not specified by SM5SEQS procedures. For example, if you define a collating sequence that includes digits and letters but not any of the special characters (such as commas, periods, slashes), a call to SM5SEQR includes the special characters in your collating sequence. The special characters all collate equally. The SM5SEQR procedure can only be called once for each call to SM5SEQN; the SM5SEQR procedure can be called anywhere in the SM5SEQx series.

If the SM5SEQA procedure is called, characters in the same value step are altered on output to the first character in the step. The following example alters all asterisks and ampersands to slashes:

```
    CALL SM5SEQS('/','*','&')
    CALL SM5SEQA('YES')
```

The SM5SEQA procedure can only be called once for each call to SM5SEQN; the SM5SEQA procedure can be called anywhere in the SM5SEQx series.

Your collating sequence specification is terminated by any procedure call other than SM5SEQS, SM5SEQR, or SM5SEQA. You can define more than one collating sequence by calling a separate series of SM5SEQx procedures for each collating sequence. The name of each collating sequence must be unique. If no SM5SEQx procedures are called, a nonstandard collating sequence is not defined.

SM5ST

Execution of the SM5ST procedure specifies an integer variable that is set to a value representing the highest level of error that occurred during the sort or merge. The format of the SM5ST call is shown in figure 5-15.

```
CALL SM5ST(variable)
```

variable	Symbolic name of an integer variable; 1 to 7 letters or digits, beginning with a letter.

Figure 5-15. SM5ST Format

The error level that is represented by the value returned to the SM5ST procedure is shown in table 5-3. For example if a 3 is returned to the SM5ST procedure, a fatal error occurred during the sort or merge. If you call the SM5ST procedure, Sort/Merge does not abort if a catastrophic error occurs before any data records are input. However, if Sort/Merge calls another product and an unrecoverable error results, an abnormal job termination will occur. For example, if Sort/Merge calls a product, such as Common Memory Manager (CMM), and CMM processing results in an error, a termination will occur.

TABLE 5-3. ERROR LEVEL CODES, SM5ST CALL

Error Level	Code
No errors	0
Trivial	10
Warning	20
Fatal	30
Catastrophic	40

SM5SUM

Execution of the SM5SUM procedure specifies fields that are to be summed in records with equal key values. Input files for a merge with summing must be presorted and presummed. As many as 255 bytes can be summed. The records with all key fields equal are combined into one new record, and the other records with equal keys are deleted. The one resultant record contains the key fields and the fields that are the sums of the specified fields from all the records with equal keys; a data field that is not a key field or a sum field is set to the corresponding field from one of the old records. The format of the SM5SUM call is shown in figure 5-16.

CALL SM5SUM(first,length,'type',rep)

first	First byte or bit of the sum field.
length	Number of bytes or bits in the sum field.
'type'	Name of a numeric data format.
rep	Number of fields to be summed.

Figure 5-16. SM5SUM Format

All the fields to be summed must contain numeric data; otherwise, the contents of the new field are undefined. The fields described as sum fields cannot also be key fields.

The SM5SUM procedure call specifies the first byte or bit of the sum field, the length of the sum field in bytes or bits, the type of data in the sum field (the name of a numeric data format except REAL), and the number of fields to be summed. See section 2 for a list of numeric data formats.

If you specify multiple fields with one SM5SUM procedure, the fields must be consecutive, be the same length, and contain the same type of numeric data. Sum fields cannot overlap one another. If a sum field contains no data because a record is short, the sum for that field is undefined.

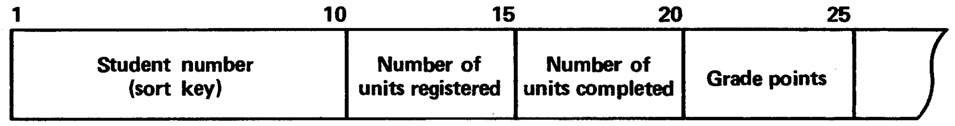
Figure 5-17 shows an example of an SM5SUM procedure call. Part A of figure 5-17 shows the layout of the record containing fields to be summed. Part B shows three SM5SUM procedure calls, each describing one field to be summed. Part C shows one equivalent SM5SUM procedure call that describes the three consecutive fields to be summed.

If the summing of the specified fields results in new numeric values that do not fit into the sum fields, a fatal error results. One of the records that caused the overflow is deleted as usual, but the sum field contents in the remaining record are undefined. A diagnostic message indicating which field caused the overflow is issued, and the job is terminated at the end of the sort (unless you call the SM5ST procedure).

You can call the SM5SUM procedure as many as 100 times during a sort or merge to specify different sum fields. If the SM5SUM procedure is not called, records with equal key values are not automatically combined into single records. You cannot call both the SM5SUM procedure and the SM5OWN5 procedure in the same sort or merge.

You cannot specify both the SM5RETA and the SM5SUM procedure in the same sort or merge. If you do specify both, a catastrophic error occurs.

A. Record Layout


1
10
15
20
25
Student number (sort key)
Number of units registered
Number of units completed
Grade points

B. SM5SUM Procedure Calls

```
CALL SM5SUM(11,5,'INTEGER',1)
CALL SM5SUM(16,5,'INTEGER',1)
CALL SM5SUM(21,5,'INTEGER',1)
```

C. SM5SUM Procedure Call

```
CALL SM5SUM(11,5,'INTEGER',3)
```

Figure 5-17. SM5SUM Procedure Call Example

SM5END

Execution of the SM5END procedure terminates your sort or merge specification and initiates Sort/Merge processing. The format of the SM5END call is shown in figure 5-18. The SM5END procedure must be the last called for any one sort or merge.

```
CALL SM5END
```

Figure 5-18. SM5END Format

COBOL INTERFACE 6

Two options are available for using Sort/Merge 5 from COBOL 5: a control statement option when you use the COBOL SORT or MERGE statement and direct calls. This section describes these two options.

COBOL5 CONTROL STATEMENT OPTION

Sort/Merge is used by COBOL when the COBOL SORT or MERGE statement is included in your program. You can select Sort/Merge 5 by means of a COBOL5 control statement option. If you specify SORT5 in your COBOL5 control statement, Sort/Merge 5 is used when the SORT or MERGE statement in your program executes. Figure 6-1 shows a sample COBOL5 control statement that includes the SORT5 option. See the COBOL 5 reference manual for information on coding the SORT or MERGE statement in a program.

```
COBOL5,I=PROGA,SORT5,L=REPTO.
```

Figure 6-1. COBOL5 Control Statement Example

DIRECT CALLS

Direct calls from a COBOL 5 program to the Sort/Merge procedures described in section 5 can be made with ENTER FTN5 SM5xxx statements. Calls to the procedures cannot be mixed with the SORT or MERGE statement; the procedures cannot be called in a COBOL input or output procedure, and the SORT or MERGE statement cannot be executed in a series of direct calls (before SM5END is called). See the COBOL 5 reference manual for information on coding ENTER statements.

All files to be sorted or merged must be sequential files. The default file characteristics are block type C, record type Z, and full length of 150 characters or less. For files with other block types and record types, you must include CYBER Record Manager (CRM) FILE control statements in the control statement section of your job (see the CRM Basic Access Methods reference manual).

OWNCODE ROUTINES 7

You can write a routine to insert, substitute, modify, or delete input and output records during Sort/Merge processing. Your routine, called an owncode routine, is executed each time the sort or merge reaches a certain point. Sort/Merge passes a record to the owncode routine, which processes the record. When the record is returned to Sort/Merge from the owncode routine, Sort/Merge processes the record according to an option specified by the owncode routine. Owncode routines can also supply the records to be sorted; when Sort/Merge is ready for a record, the owncode routine passes a record to Sort/Merge.

Owncode routines are routines written in FORTRAN (subroutine subprograms) or some other language that have been compiled and saved in relocatable binary form. The file containing your owncode routines is automatically loaded during Sort/Merge processing. This section describes specifying owncode routines and the available processing options.

SPECIFYING OWNCODE ROUTINES

To specify owncode routines when you are using control statement calls to Sort/Merge, you use the OWNF parameter. The OWNF parameter gives Sort/Merge the name of the file containing your owncode routines; the file is loaded by Sort/Merge by means of the user call loader so that the routines are available during Sort/Merge processing. The names of your owncode routines are specified with the OWNn parameter; n is an integer from 1 through 5 that tells Sort/Merge at which point in processing the routine is executed. You must supply a separate subroutine for each OWNn specified.

When you are using procedure calls to Sort/Merge, you call SM5OWNn to specify the name of an owncode routine and to indicate at which point in processing the routine is executed. Owncode routines must be loaded along with the Sort/Merge procedures. In a FORTRAN program, you must name the owncode routines in an EXTERNAL statement. You cannot call SM5OWNn from a COBOL program because there is no way (other than CALL or ENTER) to tell COBOL the name of an external subroutine. You can write a subroutine in FORTRAN or some other language to call the owncode routine.

Owncode exits 1 and 2 can be taken from a sort only; exits 3, 4, and 5 can be taken from a sort or merge.

OWNCODE ROUTINE PARAMETERS

Owncode routines are called with parameters. Up to five parameters are passed between Sort/Merge and your owncode routine. Parameters are passed at the point in processing at which your owncode routine is executed. Figure 7-1 shows the format for naming an owncode routine in a control statement and in a procedure call, and the subroutine statement format, which is a standard FORTRAN subroutine statement.

```
OWNn=proc          CALL SM5OWNn(proc)

SUBROUTINE proc(return_code,reca,rla[,recb,rlb])
```

Figure 7-1. Owncode Routine Call and Subroutine Statement Formats

If you specify an owncode 1, owncode 2, owncode 3, or owncode 4 routine, the return_code, reca, and rla parameters are passed by Sort/Merge. If you specify an owncode 5 routine, the recb and rlb parameters are passed in addition to the other three parameters. The parameters are described in the following paragraphs.

The return_code parameter is passed by Sort/Merge to an owncode routine as an integer with value 0. The return_code parameter can be altered by the owncode routine to the integer value 1, 2, or 3. The value returned to Sort/Merge by this parameter indicates a specific action to be taken by Sort/Merge. A return_code value that is not defined causes a catastrophic error. The meanings of the various return_codes are discussed later in this section.

The reca parameter is an otherwise empty array that is used to pass the current record; except for the current record, the contents of reca are undefined. The rla parameter passes a value that is the number of characters in the current record being sorted or merged.

The recb and rlb parameters are used when processing two records with equal keys. The recb parameter passes the second record; the rlb parameter passes a value that is the number of characters in the second record.

The allowed length of records passed to an owncode routine depends on how you have specified record length. If you have specified the OWNFL parameter or have called the SM5OFL procedure, the number of characters in the current record must be the same as the OWNFL or SM5OFL value. Otherwise, the maximum record length is determined as the largest MRL or FL FIT field in a CYBER Record Manager FILE control statement, or the largest value from the OWNMRL parameter or the SM5OMRL procedure; all records must contain between one and this maximum record length number of characters, inclusive. An rla or rlb parameter value that does not correspond to a record specification causes a catastrophic error.

The reca, rla, recb, and rlb parameters can be altered by an owncode routine; the routine can pass a different record back to Sort/Merge in reca or recb, and the number of characters in the record can be different. The parameters must be passed from an owncode routine in order: return_code, reca, rla, recb, and rlb. The arrays reca and recb can be any type of array.

Table 7-1 summarizes the owncode routines and the parameters passed. Figure 7-2 illustrates record movement from Sort/Merge to an owncode routine and back to Sort/Merge.

OWNCODE 1: PROCESSING INPUT RECORDS

The OWN1 parameter or the SM5OWN1 procedure call names an owncode 1 routine that is executed in one of two instances.

If you have specified input files with the FROM parameter or the SM5FROM procedure call, the owncode 1 routine is executed after reading each record. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, reca contains the record, and rla is the record length in characters. After owncode processing of the record, control returns to Sort/Merge, which processes the record passed back in reca according to the return_code value set by the owncode 1 routine (the record passed back to Sort/Merge in reca can be different from the record passed to the routine). The return_code value and the associated processing performed by Sort/Merge can be as follows:

0 The record passed back to Sort/Merge in reca is sorted.

1 The record passed back to Sort/Merge in reca is deleted.

2 An additional record is inserted into the sort. The record in reca is entered into the sort, and the owncode 1 routine is executed again with reca and rla set to the record that just entered the sort.

3 Input from the current file is terminated. The record in reca and remaining records in the file are not sorted. If more input files are specified, records are read from the next input file. The owncode 1 routine is executed after reading each record from this next file.

TABLE 7-1. OWNCODE ROUTINE SUMMARY

Routine Type	Processing	Parameters Passed					Return_code Value
		Return_code	reca	rla	recb	rlb	
Owncode 1	Input records	x	x	x			0 Sort current record. 1 Delete current record. 2 Insert new record. 3 Terminate input from current file.
Owncode 2	Input files	x	x	x			0 Begin processing next input file. 1 Insert new record.
Owncode 3	Output records	x	x	x			0 Modify current record. 1 Delete current record. 2 Create new record. 3 Terminate output.
Owncode 4	Output files	x	x	x			0 Sort or merge is complete. 1 Insert new record.
Owncode 5	Equal keys	x	x	x	x	x	0 Retain both records. 1 Replace both records with new record.

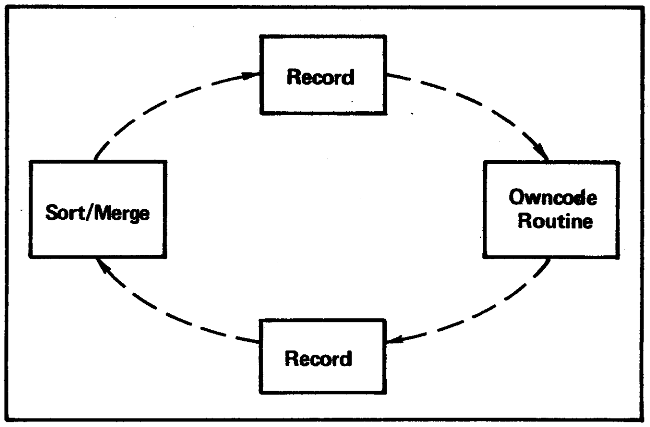


Figure 7-2. Owncode Routine Record Flow

If you have not specified any input files, the owncode 1 routine is executed when Sort/Merge is ready for another record to process. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, reca is an empty array with room for the largest record, and rla is 0. When control is returned to Sort/Merge from the owncode 1 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

0	The record passed back to Sort/Merge in reca is sorted.
2	An additional record is inserted into the sort. The record in reca is entered into the sort, and the owncode 1 routine is executed again with reca and rla set to the record that just entered the sort.
3	Input is terminated.

OWNCODE 2: PROCESSING INPUT FILES

The OWN2 parameter or the SM5OWN2 procedure call names an owncode 2 routine that is executed in one of two instances.

If you have specified input files with the FROM parameter or the SM5FROM procedure call, the owncode 2 routine is executed after input from a file has terminated (input terminates when end-of-section is found on file INPUT, when end-of-file is found on any other file, or when an owncode 1 routine passes a return_code value of 3 to Sort/Merge). The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, and reca and rla are passed as a null record; reca is an empty array, and record length is 0. When control is returned to Sort/Merge from the owncode 2 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

0	Processing of the next input file, if any, begins.
1	An additional record is inserted into the sort after the last record. The record inserted is the first rla characters in reca, which have been provided by the routine. The owncode 2 routine is executed again.

If you have not specified any input files, the owncode 2 routine is executed after an owncode 1 routine has terminated input. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, and reca and rla are passed as a null record. When control is returned to Sort/Merge from the owncode 2 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

0	Signals the end of input.
1	An additional record is inserted into the sort after the last record. The record inserted is the first rla characters in reca, which have been provided by the routine. The owncode 2 routine is executed again.

OWNCODE 3: PROCESSING OUTPUT RECORDS

The OWN3 parameter or the SM5OWN3 procedure call names an owncode 3 routine that is executed in one of two instances.

If you have specified an output file with the TO parameter or the SM5TO procedure call, the owncode 3 routine is executed before writing a record to the output file. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, reca is the record, and rla is the record length in characters. After owncode processing of the record, control returns to Sort/Merge, which processes the record passed back in reca according to the return_code value set by the owncode 3 routine. The return_code value and the associated processing performed by Sort/Merge can be as follows:

0	The record passed back to Sort/Merge in reca is written to the output file.
1	The record passed back to Sort/Merge in reca is deleted.
2	An additional record is written to the output file. The record in reca is written out, and the owncode 3 routine is executed again with reca and rla set to the original record.
3	Output from the current file is terminated. The record in reca is not written out. If an owncode 4 routine is specified, it is executed; otherwise, the sort or merge is terminated.

If you have not specified an output file, the owncode 3 routine is executed when a record is ready for output; sorted or merged records are processed by the routine. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, reca is the record, and rla is the record length in characters. When control is returned to Sort/Merge from the owncode 3 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

1 The record passed back to Sort/Merge in reca is deleted.

3 Output is terminated. If an owncode 4 routine is specified, it is executed; otherwise, the sort or merge is terminated.

OWNCODE 4: PROCESSING OUTPUT FILES

The OWN4 parameter or the SM5OWN4 procedure call names an owncode 4 routine that is executed in one of two instances.

If you have specified an output file with the TO parameter or the SM5TO procedure call, the owncode 4 routine is executed after the last record has been written to the output file. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, and reca and rla are passed as a null record. When control is returned to Sort/Merge from the owncode 4 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

0 The sort or merge is completed.

1 An additional record is inserted after the last record. The record inserted is the first rla characters in reca. The owncode 4 routine is executed again.

If you have not specified an output file, the owncode 4 routine is executed after an owncode 3 routine has terminated output. The return_code, reca, and rla parameters are passed to the routine by Sort/Merge. The return_code is 0, and reca and rla are passed as a null record. When control is returned to Sort/Merge from the owncode 4 routine, the return_code value and the associated processing performed by Sort/Merge can be as follows:

0 The sort or merge is completed.

1 This value is not allowed since no output file has been specified.

OWNCODE 5: PROCESSING RECORDS WITH EQUAL KEYS

The OWN5 parameter or the SM5OWN5 procedure call names an owncode 5 routine that is executed when two records with equal key values are encountered during a sort or merge (when you have defined your own collating sequence, key values can be equal even if key characters are not identical). The return_code, reca, rla, recb, and rlb parameters are passed to the routine by Sort/Merge. The return_code is 0; reca and recb contain the first and second records, respectively, and rla and rlb contain the record lengths in characters of the first and second records, respectively. After owncode 5 routine processing of the two records, control is returned to Sort/Merge, which processes the records according to the return_code value set by the owncode 5 routine. The return_code value and the associated processing performed by Sort/Merge can be as follows:

0 The first rla characters of reca are accepted as the first record; the first rlb characters of recb are accepted as the second record (the records and record lengths passed back to Sort/Merge can be different from the records and record lengths passed to the owncode routine).

1 One duplicate record is deleted. The other record is replaced with the first rla characters of reca.

If you specify RETAIN=YES or call the SM5RETA procedure in a sort with an owncode 5 routine, the record that entered the sort first is passed to the owncode routine as reca; otherwise, either of the two records with equal keys could be passed to the routine as reca. The owncode 5 routine can control the order in which the two records are returned to Sort/Merge. The record returned to Sort/Merge as reca is entered into the sort before the record returned as recb. An owncode 5 routine cannot be specified in the same sort as a SUM parameter or SM5SUM procedure.

EXAMPLES

This section contains examples of sorts and merges. The first example is a simple control statement sort on one key. The same sort is illustrated using the interactive dialog. A control statement sort on multiple keys, a control statement merge, directive file creation and use, SUM parameter use, and a FORTRAN program with Sort/Merge procedure calls and an owncode routine are also included.

CONTROL STATEMENT SORT ON ONE KEY

Figure 8-1 shows the record layout in a university student file named UNIVER. Each record includes the last name and first and middle initials, the student number, the date of birth, the field of study, the grade point average, and a code representing class (freshman, sophomore, junior, senior); all fields are written with character data. The file is maintained with the student number as the major key. Records are normally ordered in ascending order according to the student number as shown in figure 8-2 with file UNIVER.

Figure 8-3 shows the job structures for sorting file UNIVER to give an alphabetic list of students. Under both NOS and NOS/BE operating systems, the job statements and accounting information are followed by ATTACH commands to assign the file to the job. CYBER Record Manager (CRM) FILE control statements are included, because the input and output records contain only 38 characters each, considerably less than the default of 150. The SORT5 control statement calls for records from UNIVER to be sorted on a key that occupies character positions 1 through 10 in each record according to the default ASCII6 collating sequence in ascending order. Sorted records are written to file SRTOUT, which is created as a local file during the sort. File SRTOUT is then copied to file OUTPUT with a right shift, because the first character in the print line is reserved for carriage control. Figure 8-4 shows the output from the job.

```
WALLACE    S T  87366110255ENGIR    2861
JOHNSON    M J  90248063051MATH     2253
SANDERS    G R  99855022858BUS      3011
NEECE      M L  99911121358ART      2291
TERRELL    T H  99998040356ENG      3861
OKADA      N A 100103111750UNDEC    2225
REYES      S L 100246031558ANTHRO   3341
SUGARMAN   B T 100528070457SOC      3501
PHILLIPS   A D 100531121158EDU      2112
KRUTZ      S T 100532010353POLISCI  1981
•
•
•
WARNES     D V 102116060861POLISCI  2814
CARLSON    M K 102126022355ENGIR    3454
FUHRMAN    L W 102212111859CHEM     3204
MCMAHON    M C 102223061260ENG      2784
JUNG       G D 102301052561PHYSED   2214
POPOVICH   H W 102311100961BUS      2434
JONES      J A 102318081555EDU      2844
PEAKE      G A 102326111960JOURN    3024
BRISCOE    J H 102343121157ENVIRO   2544
HORNE      D N 102377070861COMPSCI  3894
```

Figure 8-2. File UNIVER

SORT SPECIFICATION USING INTERACTIVE DIALOG

Figure 8-5 shows the interactive dialog used to specify the same sort of file UNIVER on the last name. The input file is assigned to the job with an ATTACH command under NOS or a FETCH command under NOS/BE. CRM FILE control statements are entered for the input and output files. The dialog is invoked by typing DIALOG=YES. Sorted records are written to file SORTOUT, which Sort/Merge creates as a local file. File SORTOUT can be printed or displayed at the terminal; SORTOUT can also be routed to a printer or saved as a permanent file.

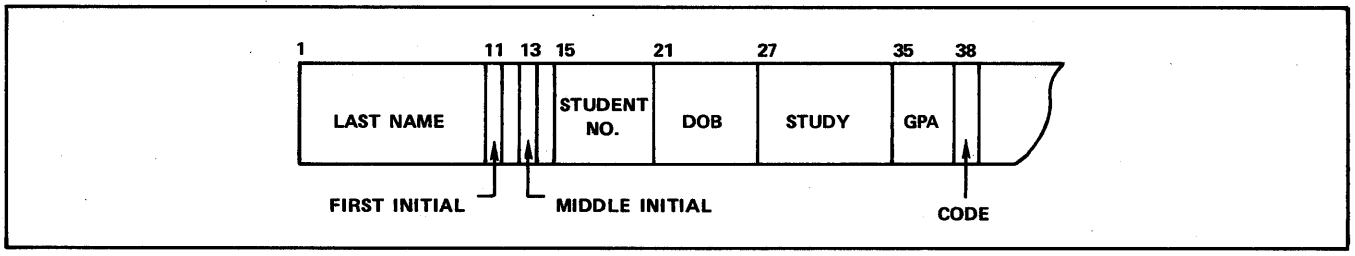


Figure 8-1. File UNIVER Record Layout

```
NOS Operating System

Job statement
USER statement
CHARGE statement
ATTACH,UNIVER.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(SRTOUT,BT=C,RT=Z,FL=38)
SORT5.FROM=UNIVER,TO=SRTOUT,KEY=1. .10
COPYSBF,SRTOUT,OUTPUT.
EOI

NOS/BE Operating System

Job statement
ACCOUNT statement
ATTACH,UNIVER,ID=PUBS.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(SRTOUT,BT=C,RT=Z,FL=38)
SORT5.FROM=UNIVER,TO=SRTOUT,KEY=1. .10
COPYSBF,SRTOUT,OUTPUT.
EOI
```

Figure 8-3. Job Structure for Sorting File UNIVER on Last Name

```
BARTLETT   S S 100800100957ART      2735
BILLINGS   C Y 101579111855MUS      2965
BRISCOE    J H 102343121157ENVIRO   2544
CARLSON    M K 102126022355ENGIR    3454
CHARLES    S H 101418032459ANTHRO   2453
CLARK      D N 101400102954ECON     3782
CLARK      D V 101023101956ENG      2083
COCHRAN    G L 100725111857BIO      3011
DAVIES     E D 100812080656JOURN    2031
DAVIS      D A 100972071650ENG      3541
●
●
●
WALLIN     G E 101056041659POLISCI 3151
WARNES     D V 102116060861POLISCI 2814
WILSON     W L 101967010261MATH     3454
WONG       S T 101001012755PSYCH    2152
WOO        R M 101315100159BUS      3223
WOODSTOCK  C T 101497030160CHEM     3483
YEH        F L 102005120645ART      2764
YOST       D L 100880111158ENG      2582
ZEITZ      F K 100963111858MATH     2612
ZIMMERS    C A 101075063059MATH     2992
```

Figure 8-4. File UNIVER Sorted on Last Name

```
/attach,univer
/file(univer,bt=c,rt=z,fl=38)
FILE(UNIVER,BT=C,RT=Z,FL=38)
/file(sortout,bt=c,rt=z,fl=38)
FILE(SORTOUT,BT=C,RT=Z,FL=38)
/sort5.dialog=yes
 ARE YOU SURE THAT ALL YOUR FILES (IF ANY) ARE EITHER
 STANDARD UNIT RECORD FORMAT OR HAVE HAD FILE CONTROL
 STATEMENTS SPECIFIED FOR THEM
? yes
 DO YOU HAVE JUST ONE INPUT FILE AND NO OWN1 OR OWN2 SUBROUTINES
? yes
 WHAT IS THE NAME OF YOUR INPUT FILE
? univer
 DO YOU HAVE ANY OWN3 OR OWN4 SUBROUTINES
? no
 WHAT IS THE NAME OF YOUR OUTPUT FILE
? sortout
 THE STANDARD COLLATING SEQUENCE IS ASCII6.
 HERE IS THE WHOLE ASCII6 COLLATING SEQUENCE LISTED
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
 DO ALL OF YOUR KEYS CONSIST OF CHARACTER DATA TO BE SORTED
 ACCORDING TO THE ASCII6 COLLATING SEQUENCE
? yes
 ASCENDING ORDER OF THE ASCII6 COLLATING SEQUENCE MEANS THE
 NORMAL ORDER.
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
 DESCENDING ORDER MEANS THE REVERSE ORDER.
 _^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#"!
 DO YOU WANT ALL YOUR KEYS SORTED IN ASCENDING ORDER
? yes
 HOW MANY KEY FIELDS DO YOU HAVE
? 1
 THE LEFT-MOST CHARACTER OF A RECORD IS NUMBER 1.
 FOR THE MOST IMPORTANT KEY FIELD, WHAT IS THE NUMBER
 OF THE LEFT MOST CHARACTER
? 1
 HOW MANY CHARACTERS ARE IN THE MOST IMPORTANT KEY FIELD
? 10
 DO YOU WANT RECORDS WITH EQUIVALENT KEY VALUES TO BE WRITTEN
 IN THE SAME ORDER THEY ARE READ
? no
 GIVE AN ESTIMATE OF THE MAXIMUM NUMBER OF RECORDS TO BE SORTED
? 75
 THANK YOU. SORT/MERGE NOW BEGINS.
SORT5.DIALOG=YES
/
```

Figure 8-5. Interactive Dialog Example

CONTROL STATEMENT SORT ON MULTIPLE KEYS

Figure 8-6 shows the job structures for sorting file UNIVER on three keys. Under both NOS and NOS/BE, the job statements and accounting information are followed by ATTACH commands to assign the file to the job. CRM FILE control statements for the input and output files are included. The SORT5 control statement calls for records to be first sorted on the field of study, which occupies character positions 27 through 34 in each record. Records with equal keys for the major key are then sorted on the class code, which is a 1-character field in position 38. The third key sorts students with the same field of study and class by the last name.

NOS Operating System

```
Job statement
USER statement
CHARGE statement
ATTACH,UNIVER.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(SRTOUT,BT=C,RT=Z,FL=38)
SORT5.FROM=UNIVER,TO=SRTOUT, . . .
.KEY=((27. .34),(38,1),(1. .10))
COPYSBF,SRTOUT,OUTPUT.
EOI
```

NOS/BE Operating System

```
Job statement
ACCOUNT statement
ATTACH,UNIVER,ID=PUBS.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(SRTOUT,BT=C,RT=Z,FL=38)
SORT5.FROM=UNIVER,TO=SRTOUT, . . .
.KEY=((27. .34),(38,1),(1. .10))
COPYSBF,SRTOUT,OUTPUT.
EOI
```

Figure 8-6. Job Structure for Control Statement Sort on Multiple Keys

Figure 8-6 illustrates continuing the SORT5 control statement beyond one line. The first line of the control statement is terminated with three periods; the second line begins with one period. Sorted records are written to the file SRTOUT, which is then copied to file OUTPUT with a right shift, because the first character in the print line is reserved for carriage control. Figure 8-7 shows the output from the job.

```
REYES     S L 100246031558ANTHRO   3341
MAYER     M I 100991122359ANTHRO   2882
CHARLES   S H 101418032459ANTHRO   2453
MARTIN    R C 100955082157ART      2891
NEECE     M L  99911121358ART      2291
NAKAMURA  S L 101529051260ART      2594
YEH       F L 102005120645ART      2764
BARTLETT  S S 100800100957ART      2735
COCHRAN   G L 100725111857BIO      3011
HOYO      J C 101925103060BIO      3014
•
•
•
KRUTZ     S T 100532010353POLISCI  1981
WALLIN    G E 101056041659POLISCI  3151
WARNES    D V 102116060861POLISCI  2814
WONG      S T 101001012755PSYCH    2152
LANGDON   M A 101754080549PSYCH    2013
LASEUR    P T 100678042256PSYCH    2233
SUGARMAN  B T 100528070457SOC      3501
SMITH     F R 101062120758SOC      2913
DOUGLAS   M L 101325071558UNDEC    2585
OKADA     N A 100103111750UNDEC    2225
```

Figure 8-7. File UNIVER Sorted on Multiple Keys

CONTROL STATEMENT MERGE

Figure 8-8 shows two files (UNIVER, ADDSTU) ordered according to the student number. The two files can be merged because they are sorted on the same key. Figure 8-9 shows the job structures to merge the two files. Under both NOS and NOS/BE, the two files are attached. A new file (NEWU) is created as a permanent file to which merged records are written; under NOS, the DEFINE command creates an empty direct access permanent file. Under NOS/BE, the REQUEST command assigns NEWU to a permanent file device; the CATALOG command then makes the file permanent. CRM FILE control statements for the input files and the merged file are included.

A. File UNIVER

```
WALLACE   S T  87366110255ENGIR    2861
JOHNSON   M J  90248063051MATH     2253
SANDERS   G R  99855022858BUS      3011
NEECE     M L  99911121358ART      2291
TERRELL   T H  99998040356ENG      3861
OKADA     N A 100103111750UNDEC    2225
REYES     S L 100246031558ANTHRO   3341
SUGARMAN  B T 100528070457SOC      3501
PHILLIPS  A D 100531121158EDU      2112
KRUTZ     S T 100532010353POLISCI  1981
•
•
•
WARNES    D V 102116060861POLISCI  2814
CARLSON   M K 102126022355ENGIR    3454
FUHRMAN   L W 102212111859CHEM     3204
MCMAHON   M C 102223061260ENG      2784
JUNG      G D 102301052561PHYSED   2214
POPOVICH  H W 102311100961BUS      2434
JONES     J A 102318081555EDU      2844
PEAKE     G A 102326111960JOURN    3024
BRISCOE   J H 102343121157ENVIRO   2544
HORNE     D N 102377070861COMPSCI  3894
```

B. File ADDSTU

```
QUINTERA  L S  90154101253BIO      3451
KING      M L 100012090848BUS      2431
ANDRUS    J R 100478042855JOURN    2121
UNGERMAN  J M 100933120356PHYSED   3012
KLEIN     S A 100987051260ENGIR    2762
GIBSON    K R 101728070160MATH     2012
IRVING    W R 101750111855ENG      3943
ALLEN     M G 102056012561LNGUIS   3854
GREENWOOD M R 102168101961EDU      2264
ANDERSEN  C R 102308032160POLISCI  2544
EBERHARD  N I 102320061158BUS      3014
GOMEZ     J R 102379022260COMPSCI  2984
```

Figure 8-8. Input Files for Control Statement Merge

```
NOS Operating System

Job statement
USER statement
CHARGE statement
ATTACH,UNIVER,ADDSTU.
DEFINE,NEWU.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(ADDSTU,BT=C,RT=Z,FL=38)
FILE(NEWU,BT=C,RT=Z,FL=38)
MERGE.FROM=(UNIVER,ADDSTU),TO=NEWU,KEY=15. .20
COPYSBF,NEWU,OUTPUT.
EOI

NOS/BE Operating System

Job statement
ACCOUNT statement
ATTACH,UNIVER,ID=PUBS.
ATTACH,ADDSTU,ID=PUBS.
REQUEST,NEWU,*PF.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FILE(ADDSTU,BT=C,RT=Z,FL=38)
FILE(NEWU,BT=C,RT=Z,FL=38)
MERGE.FROM=(UNIVER,ADDSTU),TO=NEWU,KEY=15. .20
CATALOG,NEWU,ID=PUBS,XR=CHNG.
COPYSBF,NEWU,OUTPUT.
EOI
```

Figure 8-9. Job Structure for Control Statement Merge

The MERGE control statement names the two input files and the new output file. The key on which the files are merged is the student number (the field on which the files are presorted). The merged file is copied to file OUTPUT with a right shift, because the first character in the print line is reserved for carriage control. Figure 8-10 shows the output from the job.

```
WALLACE    S T   87366110255ENGIR    2861
QUINTERA   L S   90154101253BIO      3451
JOHNSON    M J   90248063051MATH     2253
SANDERS    G R   99855022858BUS      3011
NEECE      M L   99911121358ART      2291
TERRELL    T H   99998040356ENG      3861
KING       M L  100012090848BUS      2431
OKADA      N A  100103111750UNDEC    2225
REYES      S L  100246031558ANTHRO   3341
ANDRUS     J R  100478042855JOURN    2121
•
•
•
MCMAHON    M C  102223061260ENG      2784
JUNG       G D  102301052561PHYSED   2214
ANDERSEN   C R  102308032160POLISCI  2544
POPOVICH   H W  102311100961BUS      2434
JONES      J A  102318081555EDU      2844
EBERHARD   N I  102320061158BUS      3014
PEAKE      G A  102326111960JOURN    3024
BRISCOE    J H  102343121157ENVIRO   2544
HORNE      D N  102377070861COMPSCI  3894
GOMEZ      J R  102379022260COMPSCI  2984
```

Figure 8-10. Merged File NEWU

CREATING AND USING A DIRECTIVE FILE

Figure 8-11 shows a portion of a terminal session in which a directive file is created. In the first part of figure 8-11, the directive file is created using XEDIT under NOS; (CR) denotes carriage return. In the second part of the figure, the directive file is created using EDITOR under NOS/BE.

Under NOS, XEDIT is entered after login, and the directive file to be created is given the name DIRFIL. The directive file statements begin with the word SORT followed by a comma. The statements specify a user-defined collating sequence named MYSEQ. The first SEQS parameter specifies ten steps from 0 to 9; the other SEQS parameter specifies one step consisting of hyphens, blanks, and slashes. The SEQA parameter specifies that blanks and slashes are to be output as hyphens. The directive file is then reviewed (still using XEDIT) to make sure the file is accurate. The file DIRFIL is saved as an indirect access permanent file.

Under NOS/BE, EDITOR is entered after login. The statements of the directive file are identical to the statements entered under NOS, and the directive file is then reviewed using the LIST command. Using the SAVE command, DIRFIL is made a local file without sequence numbers; the CATALOG command saves the file as a permanent file.

Figure 8-12 shows a portion of a NOS terminal session in which a file named NUMFIL (figure 8-13) is sorted according to the MYSEQ collating sequence. The directive file and the input file are assigned to the job with GET commands for indirect access permanent files. CRM FILE control statements are entered for the input and output files.

The SORT5 control statement is then entered naming the input and output files, the key field, and the directive file; the name of the user-defined collating sequence is the key type. After Sort/Merge reads the directive file, an image of this file is printed or displayed at the terminal (the L parameter default is OUTPUT). When the sort is complete, the SORT5 control statement is printed or displayed at the terminal (note that NOS repeats only 48 characters of the control statement). Control is then returned to the operating system.

Figure 8-14 shows the output from the job using the directive file with the MYSEQ collating sequence. The records are in numeric order, and the separators in the number have been standardized as hyphens in all records

SUM PARAMETER USE

Figure 8-15 shows the record layout in a file named GRADES. Each record includes three numeric fields: number of units attempted, number of units completed, and grade points. Figure 8-16 shows file GRADES with multiple records for each student. Records are to be sorted according to student number and, using the SUM parameter, records with the same student number are to be combined into one record by adding together the numeric fields to give the total number of units attempted, units completed, and the total number of grade points.

NOS Operating System

```
/xedit,dirfil,c <-------------------------  Enter XEDIT in create mode; DIRFIL is the name
 XEDIT 3.1.00                               given the file to be created.
?? <--------------------------------------  CR
 INPUT <----------------------------------  Enter XEDIT input mode.
? sort,seqn=myseq         )
? sort,seqs=('0'..'9')    }
? sort,seqs=('-',' ','/') } <-------------  Directive file statements.
? sort,seqa=yes           )
? <---------------------------------------  CR
 EDIT <-----------------------------------  Enter XEDIT edit mode.
?? top <----------------------------------  Move pointer to the beginning of the file.
?? print* <-------------------------------  Print the entire file.
SORT,SEQN=MYSEQ           )
SORT,SEQS=('0'..'9')      }
SORT,SEQS=('-',' ','/')   } <-------------  Directive file statements.
SORT,SEQA=YES             )
END OF FILE
?? end,dirfil,replace <-------------------  Leave XEDIT and save DIRFIL as an indirect access
DIRFIL  REPLACED                            permanent file.
```

NOS/BE Operating System

```
COMMAND- editor <-------------------------  Enter EDITOR.
..1=sort,seqn=myseq       )
2=sort,seqs=('0'..'9')    } <-------------  Directive file statements.
3=sort,seqs=('-',' ','/') )
4=sort,seqa=yes
list,a,sup <------------------------------  List the entire file.


SORT,SEQN=MYSEQ           )
SORT,SEQS=('0'..'9')      }
SORT,SEQS=('-',' ','/')   } <-------------  Directive file statements.
SORT,SEQA=YES             )
..save,dirfil,n <-------------------------  Make edit file a local file without sequence numbers.
..end <-----------------------------------  Leave EDITOR.
COMMAND- catalog,dirfil,id=pubs <---------  Save file as a permanent file named DIRFIL.
```

Figure 8-11. Creating a Directive File

```
/get,dirfil
/get,numfil
/file(numfil,bt=c,rt=z,fl=35)
FILE(NUMFIL,BT=C,RT=Z,FL=35)
/file(seqfil,bt=c,rt=z,fl=35)
FILE(SEQFIL,BT=C,RT=Z,FL=35)
/sort5.from=numfil,to=seqfil,key=((25,11,myseq)),dir=dirfil
1
              SORT/MERGE 5.0     * LISTING OF DIRECTIVE FILE - DIRFIL
12/04.      16.45.45.
        1.       SORT,SEQN=MYSEQ
        2.       SORT,SEQS=('0'..'9')
        3.       SORT,SEQS=('-',' ','/')
        4.       SORT,SEQA=YES
  SORT/MERGE 5 -         18 RECORDS SORTED
/
```

Figure 8-12. Using a Directive File (NOS)

```
DOE         CHERYL      706-87-5430
THOMPSON    LOUISE      819 95 6445
GORDON      ROBERT      176/81/0013
WATSON      JOSEPH      091-15-1938
ANDERSON    PETER       778 21 2847
HALL        MARY        029-55-1789
SMITH       JOHN        555-70-6351
NORTH       PATRICIA    234-58-3190
ROLLINS     SARAH       770-25-8147
SHARP       DANIEL      447/46/0043
LYNCH       DENNIS      549-22-6741
SMITH       CATHERINE   387 14 9998
NOE         JANE        880-25-3058
BROWN       WILLIAM     120 59 4049
HARRIS      JEAN        229-80-7776
JONES       CHRISTOPHER 667/26/2514
CARTER      BARBARA     114 48 5112
HENDERS     GERALD      998-72-6203
```

Figure 8-13. File NUMFIL

```
HALL        MARY        029-55-1789
WATSON      JOSEPH      091-15-1938
CARTER      BARBARA     114-48-5112
BROWN       WILLIAM     120-59-4049
GORDON      ROBERT      176-81-0013
HARRIS      JEAN        229-80-7776
NORTH       PATRICIA    234-58-3190
SMITH       CATHERINE   387-14-9998
SHARP       DANIEL      447-46-0043
LYNCH       DENNIS      549-22-6741
SMITH       JOHN        555-70-6351
JONES       CHRISTOPHER 667-26-2514
DOE         CHERYL      706-87-5430
ROLLINS     SARAH       770-25-8147
ANDERSON    PETER       778-21-2847
THOMPSON    LOUISE      819-95-6445
NOE         JANE        880-25-3058
HENDERS     GERALD      998-72-6203
```

Figure 8-14. Output From Sort Using a Directive File

```
GREENWOOD M R 102168101961EDU      002002000
IRVING    W R 101750111855ENG      004004016
IRVING    W R 101750111855ENG      098095375
GREENWOOD M R 102168101961EDU      003003009
ANDRUS    J R 100478042855JOURN    005005010
ANDRUS    J R 100478042855JOURN    004000000
ALLEN     M G 102056012561LNGUIS   003003012
KLEIN     S A 100987051260ENGIR    003003006
QUINTERA  L S  90154101253BIO      003000000
IRVING    W R 101750111855ENG      003003012
ANDRUS    J R 100478042855JOURN    190173371
ANDRUS    J R 100478042855JOURN    003003009
KING      M L 100012090848BUS      005005015
ALLEN     M G 102056012561LNGUIS   005000000
ALLEN     M G 102056012561LNGUIS   025020077
ALLEN     M G 102056012561LNGUIS   004004012
ALLEN     M G 102056012561LNGUIS   005005020
KING      M L 100012090848BUS      004004016
EBERHARD  N I 102320061158BUS      005005015
EBERHARD  N I 102320061158BUS      001001000
EBERHARD  N I 102320061158BUS      004004016
KLEIN     S A 100987051260ENGIR    005005015
EBERHARD  N I 102320061158BUS      003003006
GIBSON    K R 101728070160MATH     003000000
KLEIN     S A 100987051260ENGIR    003003006
KLEIN     S A 100987051260ENGIR    135135373
ANDERSEN  C R 102308032160POLISCI 005005015
ANDERSEN  C R 102308032160POLISCI 003003009
QUINTERA  L S  90154101253BIO      180176607
UNGERMAN  J M 100933120356PHYSED   150148445
ANDERSEN  C R 102308032160POLISCI 005005010
UNGERMAN  J M 100933120356PHYSED   003003012
UNGERMAN  J M 100933120356PHYSED   003003009
KING      M L 100012090848BUS      002002006
KING      M L 100012090848BUS      178174423
ANDRUS    J R 100478042855JOURN    001001000
GREENWOOD M R 102168101961EDU      003003006
GREENWOOD M R 102168101961EDU      030030068
IRVING    W R 101750111855ENG      003003009
GIBSON    K R 101728070160MATH     145143290
GIBSON    K R 101728070160MATH     005005010
QUINTERA  L S  90154101253BIO      005005010
GOMEZ     J R 102379022260COMPSCI 003003009
GOMEZ     J R 102379022260COMPSCI 005005010
GOMEZ     J R 102379022260COMPSCI 002002008
GREENWOOD M R 102168101961EDU      005005020
QUINTERA  L S  90154101253BIO      004004012
```

Figure 8-16. File GRADES

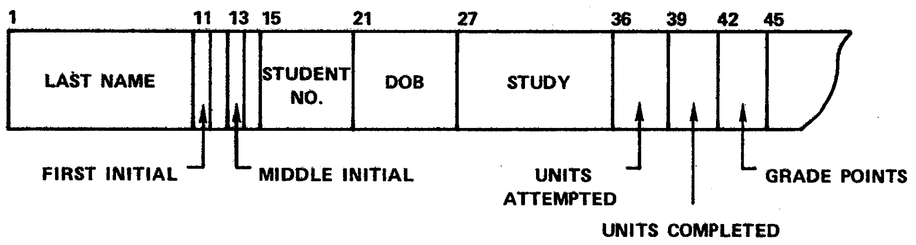


Figure 8-15. File GRADES Record Layout

Figure 8-17 shows the SORT5 control statement with the SUM parameter. The input and output files are named, and the student number (character positions 15 through 20) is specified as the sort key. The SUM parameter specifies that a 3-character numeric field of type NUMERIC_NS begins in position 36 in each record. The repetition indicator 3 specifies that three contiguous fields are to be summed. Figure 8-18 shows the output from the job. The output file contains one record for each student. The numeric fields are the totals of the units attempted, units completed, and grade points.

A SORT USING PROCEDURE CALLS

Figure 8-19 shows the job structures for performing a sort from within a FORTRAN program. Under both NOS and NOS/BE, the job statements and accounting information are followed by ATTACH commands to assign the input data file (UNIVER) and the file containing the FORTRAN source program to the job. The FORTRAN 5 compiler is called, and the program is compiled. The program is then executed (LGO).

Figure 8-20 shows the FORTRAN program containing the Sort/Merge procedure calls. File UNIVER is read, and student records with grade point average of 3.50 or better are written to an intermediate file (INT1). Sort/Merge is called to sort the file on grade point average in descending order (highest grade point average to lowest grade point average). The SM5SORT call names a result array that is dimensioned as a 16-element integer array and the first element set to 15 to receive the maximum number of sort results and statistics.

The SM5OWN1 call specifies that an owncode 1 routine named CCODE is to be executed after Sort/Merge reads each record from INT1. Records are passed to the routine by Sort/Merge. Figure 8-21 shows the routine, which is a FORTRAN subroutine. The SUBROUTINE statement names the routine and the parameters passed by Sort/Merge. Parameter RETCODE is the return_code passed as 0, REC is an array containing the record, and RL is the record length in characters. The routine converts the class code in each record to the class name. The records are returned to Sort/Merge in array REC. The return_code value is left as 0 because each record in this example is to be sorted. Sort/Merge then sorts the records to file INT2. The sorted file is returned to the FORTRAN program to be written out in a formatted report. Figure 8-22 shows the output from the job, which includes the report and the contents of the result array.

```
/get,grades
/file(grades,bt=c,rt=z,fl=44)
FILE(GRADES,BT=C,RT=Z,FL=44)
/file(sumex,bt=c,rt=z,fl=44)
FILE(SUMEX,BT=C,RT=Z,FL=44)
/sort5.from=grades,to=sumex,key=15..20,sum=((36,3,numeric_ns,3))
 SORT/MERGE 5 -        47 RECORDS SORTED
/
```

Figure 8-17. Interactive Parameter Specification for SUM Example

Total units completed

Total units attempted

Total grade points

```
QUINTERA  L S 090154101253BIO       192185629
KING      M L 100012090848BUS       189185460
ANDRUS    J R 100478042855JOURN     203182390
UNGERMAN  J M 100933120356PHYSED    156154466
KLEIN     S A 100987051260ENGIR     146146400
GIBSON    K R 101728070160MATH      153148300
IRVING    W R 101750111855ENG       108105412
ALLEN     M G 102056012561LNGUIS    042032121
GREENWOOD M R 102168101961EDU       043043103
ANDERSEN  C R 102308032160POLISCI   013013034
EBERHARD  N I 102320061158BUS       013013037
GOMEZ     J R 102379022260COMPSCI   010010027
```

Figure 8-18. Output From SUM Example

NOS Operating System

```
Job statement
USER statement
CHARGE statement
ATTACH,UNIVER,SM5EX.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FTN5,I=SM5EX,L=OUTEX.
LGO.
COPY,OUTEX,OUTPUT.
EOI
```

NOS/BE Operating System

```
Job statement
ACCOUNT statement
ATTACH,UNIVER,ID=PUBS.
ATTACH,SM5EX,ID=F.
FILE(UNIVER,BT=C,RT=Z,FL=38)
FTN5,I=SM5EX,L=OUTEX.
LGO.
COPY,OUTEX,OUTPUT.
EOI
```

Figure 8-19. Job Structure for Sort Called From a FORTRAN Program

```
      PROGRAM DLIST
      INTEGER GPA
      CHARACTER SNAME*14, MAJOR*8, CODE*1, CLASS*12
      DIMENSION IARAY(16)
      COMMON /COMN/ CODE, CLASS
      EXTERNAL CCODE
C
    1 READ(L"UNIVER",100,END=10) SNAME, MAJOR, GPA, CODE
      IF (GPA .GE. 350) WRITE(L"INT1",200) SNAME, MAJOR, GPA, CODE
      GO TO 1
C
   10 IARAY(1)=15
      CALL SM5SORT(IARAY)
      CALL SM5OFL(80)
      CALL SM5FROM('INT1')
      CALL SM5KEY(33,3,'ASCII6','D')
      CALL SM5OWN1(CCODE)
      CALL SM5TO('INT2')
      CALL SM5END
      REWIND(L"INT2")
C
      PRINT *, IARAY
      WRITE(L"OUTEX",400)
   15 READ(L"INT2",300,END=20) SNAME, MAJOR, GPA, CLASS
      WRITE(L"OUTEX",500) SNAME, MAJOR, CLASS, GPA
      GO TO 15
C
  100 FORMAT (A14,12X,A8,I3,A1)
  200 FORMAT (A14,5X,A8,5X,I3,5X,A1,39X)
  300 FORMAT (A14,5X,A8,5X,I3,5X,A12,28X)
  400 FORMAT (36X,"DEAN'S LIST" // 15X,"STUDENT",
     *     12X,"MAJOR",8X,"CLASS",12X,"GPA",65X /)
  500 FORMAT (15X,A14,5X,A8,5X,A12,5X,I3,59X)
C
   20 STOP
      END
```

Figure 8-20. FORTRAN Program With Sort/Merge Procedure Calls

```
C
      SUBROUTINE CCODE(RETCODE,REC,RL)
      INTEGER RETCODE, RL
      CHARACTER C1*6, C2*6, C3*9, C4*8, C5*12,
     *          CODE*1, CLASS*12, REC(*)*12
      COMMON /SUBCOM/ CODE, CLASS
      DATA C1,C2/'SENIOR','JUNIOR'/
      DATA C3,C4/'SOPHOMORE','FRESHMAN'/
      DATA C5/'UNCLASSIFIED'/
C
      CODE = REC(4)(5:5)
      IF (CODE .EQ. '1') THEN
         CLASS = C1
      ELSE IF (CODE .EQ. '2') THEN
         CLASS = C2
      ELSE IF (CODE .EQ. '3') THEN
         CLASS = C3
      ELSE IF (CODE .EQ. '4') THEN
         CLASS = C4
      ELSE IF (CODE .EQ. '5') THEN
         CLASS = C5
      ELSE
         PRINT *, CODE
      END IF
      REC(4)(5:12) = CLASS(1:8)
      REC(5)(1:4) = CLASS(9:12)
      RETURN
      END
```

Figure 8-21. Owncode Routine

A. Report

```
                     DEAN'S LIST

STUDENT              MAJOR       CLASS             GPA

SHIELDS    L E       COMPSCI     SENIOR            390
HORNE      D N       COMPSCI     FRESHMAN          389
TERRELL    T H       ENG         SENIOR            386
SMITH      F R       PHIL        SOPHOMORE         385
SMITH      C R       MATH        SENIOR            379
CLARK      D N       ECON        JUNIOR            378
TIEMON     H R       LNGUIS      SOPHOMORE         376
FRANKLIN   R H       PHIL        JUNIOR            370
HANSEN     R P       BUS         SOPHOMORE         358
DAVIS      D A       ENG         SENIOR            354
SUGARMAN   B T       SOC         SENIOR            350
```

B. Result Array

$\underset{\uparrow}{15}\ \underbrace{11\ 0\ 0\ 0\ 11\ 0\ 0\ 0\ 0\ 0\ 0\ 11\ 80\ 80\ 80}$

Element 1; user set.

Elements 2 thru 16; set by Sort/Merge. See table 5-1.

Figure 8-22. Output From FORTRAN Program

CHARACTER DATA INPUT, OUTPUT, AND CENTRAL MEMORY REPRESENTATION

This appendix describes the code and character sets used by host computer operating system local batch device drivers, magnetic tape drivers, and terminal communication products. Some software products assume that certain graphic or control characters are associated with specific binary code values for collating or syntax processing purposes. This appendix does not describe those associations.

All references within this manual to the ASCII character set or the ASCII code set refer to the character set and code set defined in the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to the ASCII character set do not necessarily apply to the ASCII code set. Sort/Merge uses a 63- or 64-character subset of the full ASCII set. Only characters in 6-bit or internal BCD display code should be used with Sort/Merge.

CHARACTER SETS AND CODE SETS

A character set differs from a code set. A character set is a set of graphic and/or control characters. A code set is a system of symbols used to represent each character within a character set. Characters exist outside the computer system and communication network; codes are received, stored, retrieved, and transmitted within the computer system and network.

GRAPHIC AND CONTROL CHARACTERS

A graphic character can be displayed at a terminal or printed by a line printer. Examples of graphic characters are the characters A through Z, a blank, and the digits 0 through 9. A control character initiates, modifies, or stops a control operation. An example of a control character is the backspace character, which moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, some terminals can produce a graphic representation when they receive a control character.

CODED AND BINARY CHARACTER DATA

Character codes can be interpreted as coded character data or as binary character data. Coded character data is converted from one code set representation to another as it enters or leaves the computer system; for example, data received from a terminal or sent to a magnetic tape unit is converted. Binary character data is not converted as it enters or leaves the system. Character codes are not converted when moved within the system; for example, data transferred to or from mass storage is not converted.

The distinction between coded character data and binary character data is important when reading or punching cards and when reading or writing magnetic tape. Only coded character data can be properly reproduced as characters on a line printer. Only binary character data can properly represent characters on a punched card when the data cannot be stored as display code.

The distinction between binary character data and characters represented by binary data (such as peripheral equipment instruction codes) is also important. Only such binary noncharacter data can properly reproduce characters on a plotter.

FORMATTED AND UNFORMATTED CHARACTER DATA

Character codes can be interpreted by a product as formatted character data or as unformatted character data. Formatted data can be stored or retrieved by a product in the form of the codes described for coded character data in the remainder of this appendix, or formatted data can be altered to another form during storage or retrieval; for example, 1 can be stored as a character code or as an integer value. Treatment of unformatted data by a product includes both coded character data and binary character data as described in this appendix.

NETWORK OPERATING SYSTEMS

The NOS and NOS/BE operating systems support the following character sets:

- CDC graphic 64-character set
- CDC graphic 63-character set
- ASCII graphic 64-character set
- ASCII graphic 63-character set
- ASCII graphic 95-character set

In addition, NOS supports the ASCII 128-character graphic and control set.

Each installation must select either a 64-character set or a 63-character set. The differences between the codes of a 63-character set and the codes of a 64-character set are described under Character Set Anomalies. Any reference in this appendix to a 64-character set implies either a 63- or 64-character set unless otherwise stated.

To represent its six listed character sets in central memory, NOS supports the following code sets:

- 6-bit display code
- 12-bit ASCII code
- 6/12-bit display code

To represent its five listed character sets in central memory, NOS/BE supports the following code sets:

- 6-bit display code
- 12-bit ASCII code

Under both NOS and NOS/BE, the 6-bit display code is a set of 6-bit codes from 00_8 to 77_8.

Under both NOS and NOS/BE, the 12-bit ASCII code is the ASCII 7-bit code (as defined by ANSI Standard X3.4-1977) right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 12-bit ASCII 0000_8 code from the end-of-line byte. The 12-bit codes are 0001_8 through 0177_8 and 4000_8.

Under NOS, the 6/12-bit display code is a combination of 6-bit codes and 12-bit codes. The 6-bit codes are 00_8 through 77_8, excluding 74_8 and 76_8. (The interpretation of the 00_8 and 63_8 codes is described under Character Set Anomalies in this appendix.) The 12-bit codes begin with either 74_8 or 76_8 and are followed by a 6-bit code. Thus, 74_8 and 76_8 are considered escape codes and are never used as 6-bit codes within the 6/12-bit display code set. The 12-bit codes are 7401_8, 7402_8, 7404_8, 7407_8, and 7601_8 through 7677_8. All other 12-bit codes ($74xx_8$ and 7600_8) are undefined.

CHARACTER SET ANOMALIES

The operating system input/output software and some products interpret two codes differently when the installation selects a 63-character set rather than a 64-character set. If an installation uses a 63-character set, the colon graphic character is always represented by a 63_8 display code, display code 00_8 is undefined (it has no associated graphic or punched card code), and the % graphic does not exist.

However, under NOS, if the installation uses a 64-character set, output of a 7404_8 6/12-bit display code or a 00_8 display code produces a colon. A colon can be input only as a 7404_8 6/12-bit display code. The use of undefined 6/12-bit display codes in output files produces unpredictable results and should be avoided.

Under NOS/BE, if the installation uses a 64-character set, output of a 00_8 display code produces a colon. Display code 63_8 is the colon when a 63-character set is used. The % graphic and related card codes do not exist and translations yield a blank (55_8).

Under both NOS and NOS/BE, two consecutive 00_8 codes can be confused with an end-of-line byte and should be avoided.

CHARACTER SET TABLES

The character set tables A-1 and A-2 are designed so that you can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, look up the code in the column listing the appropriate code set and then find the character on that line in the column listing the appropriate character set. To find the code that represents a character, look up the character and then find the code on the same line in the appropriate column.

Conversational Terminal Users

Table A-1 shows the character sets and code sets available to an Interactive Facility (IAF) or INTERCOM user at an ASCII code terminal using an ASCII character set. The Full ASCII Character Set table (later in this appendix) shows the octal and hexadecimal 7-bit ASCII code for each ASCII character, and can be used to convert codes from octal to hexadecimal. (Under NOS using network product software, certain Terminal Interface Program commands require specification of an ASCII code.

IAF Usage

When in normal time-sharing mode (specified by the IAF NORMAL command), IAF assumes the ASCII graphic 64-character set is used and translates all input and output to or from display code. When in ASCII time-sharing mode (specified by the IAF ASCII command), IAF assumes the ASCII 128-character set is used and translates all input and output to or from 6/12-bit display code.

The IAF user can convert a 6/12-bit code file to a 12-bit ASCII code file using the NOS FCOPY control statement. The resulting 12-bit ASCII file can be routed to a line printer but cannot be output through IAF.

IAF supports both character mode and transparent mode transmissions through the network. These transmission modes are described under Network Access Method Terminal Transmission Code Sets in this appendix. IAF treats character mode transmissions as coded character data; IAF converts these transmissions to or from either 6-bit or 6/12-bit display code. IAF treats transparent mode transmissions as binary character data; transparent mode communication between IAF and ASCII terminals using any parity setting occurs in the 12-bit ASCII code shown in table A-1.

INTERCOM Usage

By default, INTERCOM displays the ASCII graphic 64-character set and interprets all input and output as display code. Refer to the INTERCOM reference manual.

COMPASS and FORTRAN users can elect to use 12-bit ASCII code if the terminal in use supports the code set selected. BASIC users can elect to send and receive lowercase and uppercase character codes using the 12-bit ASCII code if the terminal in use supports the code set selected; BASIC represents coded character data in central memory using 6/12-bit display code in both the NOS and NOS/BE systems.

TABLE A-1. CONVERSATIONAL TERMINAL CHARACTER SETS

ASCII Graphic (64-Character Set)	ASCII Character (128-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code†	Octal 12-Bit ASCII Code	ASCII Graphic (64-Character Set)	ASCII Character (128-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code†	Octal 12-Bit ASCII Code
: colon††		00††				^ circumflex		7402	0136
A	A	01	01	0101		: colon††		7404††	0072
B	B	02	02	0102		` grave accent		7407	0140
C	C	03	03	0103		a		7601	0141
D	D	04	04	0104		b		7602	0142
E	E	05	05	0105		c		7603	0143
F	F	06	06	0106		d		7604	0144
G	G	07	07	0107		e		7605	0145
H	H	10	10	0110		f		7606	0146
I	I	11	11	0111		g		7607	1047
J	J	12	12	0112		h		7610	0150
K	K	13	13	0113		i		7611	0151
L	L	14	14	0114		j		7612	0152
M	M	15	15	0115		k		7613	0153
N	N	16	16	0116		l		7614	0154
O	O	17	17	0117		m		7615	0155
P	P	20	20	0120		n		7616	0156
Q	Q	21	21	0121		o		7617	0157
R	R	22	22	0122		p		7620	0160
S	S	23	23	0123		q		7621	0161
T	T	24	24	0124		r		7622	0162
U	U	25	25	0125		s		7623	0163
V	V	26	26	0126		t		7624	0164
W	W	27	27	0127		u		7625	0165
X	X	30	30	0130		v		7626	0166
Y	Y	31	31	0131		w		7627	0167
Z	Z	32	32	0132		x		7630	0170
0	0	33	33	0060		y		7631	0171
1	1	34	34	0061		z		7632	0172
2	2	35	35	0062		{ left brace		7633	0173
3	3	36	36	0063		| vert. line		7634	0174
4	4	37	37	0064		} right brace		7635	0175
5	5	40	40	0065		~ tilde		7636	0176
6	6	41	41	0066		NUL		7640	4000
7	7	42	42	0067		SOH		7641	0001
8	8	43	43	0070		STX		7642	0002
9	9	44	44	0071		ETX		7643	0003
+ plus	+ plus	45	45	0053		EOT		7644	0004
- minus	- minus	46	46	0055		ENQ		7645	0005
* asterisk	* asterisk	47	47	0052		ACK		7646	0006
/ slash	/ slash	50	50	0057		BEL		7647	0007
(l. paren.	(l. paren.	51	51	0050		BS		7650	0010
) r. paren.	) r. paren.	52	52	0051		HT		7651	0011
$ dollar	$ dollar	53	53	0044		LF		7652	0012
= equal to	= equal to	54	54	0075		VT		7653	0013
space	space	55	55	0040		FF		7654	0014
, comma	, comma	56	56	0054		CR		7655	0015
. period	. period	57	57	0056		SO		7656	0016
# number	# number	60	60	0043		SI		7657	0017
[l. bracket	[l. bracket	61	61	0133		DEL		7637	0177
] r. bracket	] r. bracket	62	62	0135		DLE		7660	0020
% percent††	% percent††	63††	63††	0045		DC1		7661	0021
" quote	" quote	64	64	0042		DC2		7662	0022
_ underline	_ underline	65	65	0137		DC3		7663	0023
! exclam.	! exclam.	66	66	0041		DC4		7664	0024
& ampersand	& ampersand	67	67	0046		NAK		7665	0025
' apostrophe	' apostrophe	70	70	0047		SYN		7666	0026
? question	? question	71	71	0077		ETB		7667	0027
< less than	< less than	72	72	0074		CAN		7670	0030
> grtr. than	> grtr. than	73	73	0076		EM		7671	0031
@ coml. at		74				SUB		7672	0032
\ rev. slant	\ rev. slant	75	75	0134		ESC		7673	0033
^ circumflex		76				FS		7674	0034
; semicolon	; semicolon	77	77	0073		GS		7675	0035
	@ coml. at		7401	0100		RS		7676	0036
						US		7677	0037

†Generally available only on NOS, or through BASIC on NOS/BE.

††The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in the text.

TABLE A-2. LOCAL BATCH DEVICE CHARACTER SETS

CDC Graphic (64-Character Set)	ASCII Graphic (64-Character Set)	ASCII Graphic (95-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code†	Octal 12-Bit ASCII Code	Card Keypunch Code	
						026	029
: colon††	: colon††		00††			8-2	8-2
A	A	A	01	01	0101	12-1	12-1
B	B	B	02	02	0102	12-2	12-2
C	C	C	03	03	0103	12-3	12-3
D	D	D	04	04	0104	12-4	12-4
E	E	E	05	05	0105	12-5	12-5
F	F	F	06	06	0106	12-6	12-6
G	G	G	07	07	0107	12-7	12-7
H	H	H	10	10	0110	12-8	12-8
I	I	I	11	11	0111	12-9	12-9
J	J	J	12	12	0112	11-1	11-1
K	K	K	13	13	0113	11-2	11-2
L	L	L	14	14	0114	11-3	11-3
M	M	M	15	15	0115	11-4	11-4
N	N	N	16	16	0116	11-5	11-5
O	O	O	17	17	0117	11-6	11-6
P	P	P	20	20	0120	11-7	11-7
Q	Q	Q	21	21	0121	11-8	11-8
R	R	R	22	22	0122	11-9	11-9
S	S	S	23	23	0123	0-2	0-2
T	T	T	24	24	0124	0-3	0-3
U	U	U	25	25	0125	0-4	0-4
V	V	V	26	26	0126	0-5	0-5
W	W	W	27	27	0127	0-6	0-6
X	X	X	30	30	0130	0-7	0-7
Y	Y	Y	31	31	0131	0-8	0-8
Z	Z	Z	32	32	0132	0-9	0-9
0	0	0	33	33	0060	0	0
1	1	1	34	34	0061	1	1
2	2	2	35	35	0062	2	2
3	3	3	36	36	0063	3	3
4	4	4	37	37	0064	4	4
5	5	5	40	40	0065	5	5
6	6	6	41	41	0066	6	6
7	7	7	42	42	0067	7	7
8	8	8	43	43	0070	8	8
9	9	9	44	44	0071	9	9
+ plus	+ plus	+ plus	45	45	0053	12	12-8-6
- minus	- minus	- minus	46	46	0055	11	11
* asterisk	* asterisk	* asterisk	47	47	0052	11-8-4	11-8-4
/ slash	/ slash	/ slash	50	50	0057	0-1	0-1
(left paren.	(left paren.	(left paren.	51	51	0050	0-8-4	12-8-5
) right paren.	) right paren.	) right paren.	52	52	0051	12-8-4	11-8-5
$ dollar	$ dollar	$ dollar	53	53	0044	11-8-3	11-8-3
= equal to	= equal to	= equal to	54	54	0075	8-3	8-6
space	space	space	55	55	0040	no punch	no punch
, comma	, comma	, comma	56	56	0054	0-8-3	0-8-3
. period	. period	. period	57	57	0056	12-8-3	12-8-3
≡ equivalence	# number	# number	60	60	0043	0-8-6	8-3
[left bracket	[left bracket	[l. bracket	61	61	0133	8-7	12-8-2 or 12-0†††
] right bracket	] right bracket	] r. bracket	62	62	0135	0-8-2	11-8-2 or 11-0†††
% percent††	% percent††	% percent††	63††	63††	0045	8-6	0-8-4

TABLE A-2. LOCAL BATCH DEVICE CHARACTER SETS (Contd)

CDC Graphic (64-Character Set)	ASCII Graphic (64-Character Set)	ASCII Graphic (95-Character Set)	Octal 6-Bit Display Code	Octal 6/12-Bit Display Code[†]	Octal 12-Bit ASCII Code	Card Keypunch Code	
						026	029
≠ not equal	" quote	" quote	64	64	0042	8-4	8-7
↱ concat.	_ underline	_ underline	65	65	0137	0-8-5	0-8-5[§] or 11-0
∨ logical OR	! exclamation	! exclamation	66	66	0041	11-0[§] or 11-8-2	12-8-7
∧ logical AND	& ampersand	& ampersand	67	67	0046	0-8-7	12
↑ superscript	' apostrophe	' apostrophe	70	70	0047	11-8-5	8-5
↓ subscript	? question	? question	71	71	0077	11-8-6	0-8-7
< less than	< less than	< less than	72	72	0074	12-0	12-8-4
> greater than	> greater than	> greater than	73	73	0076	11-8-7	0-8-6
≤ less/equal	@ commercial at		74			8-5	8-4
≥ greater/equal	\ reverse slant	\ rev. slant	75	75	0134	12-8-5	0-8-2
¬ logical NOT	circumflex		76			12-8-6	11-8-7
; semicolon	; semicolon	; semicolon	77	77	0073	12-8-7	11-8-6
		@ coml. at		7401	0100		
		^ circumflex		7402	0136		
		: colon		7404	0072		
		` grave accent		7407	0140		
		a		7601	0141		
		b		7602	0142		
		c		7603	0143		
		d		7604	0144		
		e		7605	0145		
		f		7606	0146		
		g		7607	0147		
		h		7610	0150		
		i		7611	0151		
		j		7612	0152		
		k		7613	0153		
		l		7614	0154		
		m		7615	0155		
		n		7616	0156		
		o		7617	0157		
		p		7620	0160		
		q		7621	0161		
		r		7622	0162		
		s		7623	0163		
		t		7624	0164		
		u		7625	0165		
		v		7626	0166		
		w		7627	0167		
		x		7630	0170		
		y		7631	0171		
		z		7632	0172		
		{ left brace		7633	0173		
		| vert. line		7634	0174		
		} right brace		7635	0175		
		~ tilde		7636	0176		

[†] Generally available only on NOS, or through BASIC on NOS/BE.

[††] The interpretation of this character or code depends on its context. Refer to Character Set Anomalies in the text.

[†††] Available for input only, on NOS.

[§] Available for input only, on NOS/BE.

Local Batch Users

Table A-2 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character set. This table also lists the code sets and card keypunch codes (026 and 029) that represent the characters.

The 64-character sets use display code as their code set; the 95-character set uses 12-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to Line Printer Output). Only 12-bit ASCII code files can be printed using the ASCII graphic 95-character set. To print a 6/12-bit display code file (usually created in IAF ASCII mode), you must convert the file to 12-bit ASCII code. To do this, the NOS FCOPY control statement must be issued. The 95-character set is represented by the 12-bit ASCII codes 0040_8 through 0176_8.

Line Printer Output

The batch character set printed depends on the print train used on the line printer to which the file is sent. The following are the print trains corresponding to each of the batch character sets:

Character Set	Print Train
CDC graphic 64-character set	596-1
ASCII graphic 64-character set	596-5
ASCII graphic 95-character set	596-6

The characters of the default 596-1 print train are listed in the table A-2 column labeled CDC Graphic (64-Character); the 596-5 print train characters are listed in the table A-2 column labeled ASCII Graphic (64-Character); and the 596-6 print train characters are listed in the table A-2 column labeled ASCII Graphic (95-Character).

If a transmission error occurs during the printing of a line, NOS prints the line again. The CDC graphic print train prints a concatenation symbol (↱) in the first printable column of a line containing errors. The ASCII print trains print an underline instead of the concatenation symbol.

If an unprintable character exists in a line (that is, a 12-bit ASCII code outside of the range 0040_8 through 0176_8), the number sign (#) appears in the first printable column of a print line and a space replaces the unprintable character.

Punched Card Input and Output

Under NOS, coded character data is exchanged with local batch card readers or card punches according to the translations shown in table A-2. As indicated in the table, additional card keypunch codes are available for input of the ASCII and CDC characters [and]. The 95-character set cannot be read or punched as coded character data.

Depending on an installation or deadstart option, NOS assumes an input deck has been punched either in 026 or 029 keypunch code (regardless of the character set in use). The alternate keypunch codes can be specified by a 26 or 29 punched in columns 79 and 80 of any job card, 6/7/9 card, or 7/8/9 card. The specified code translation remains in effect throughout the job unless it is reset by specification of the alternate code translation on a subsequent 6/7/9 card or 7/8/9 card.

NOS keypunch code translation can also be changed by a card containing a 5/7/9 punch in column 1. A blank (no punch) in column 2 indicates 026 conversion mode; a 9 punch in column 2 indicates 029 conversion mode. The conversion change remains in effect until another change card is encountered or the job ends.

The 5/7/9 card also allows literal input when 4/5/6/7/8/9 is punched in column 2. Literal input can be used to read 80-column binary character data within a punched card deck of coded character data.

Literal cards are stored with each column in a 12-bit byte (a row 12 punch is represented by a 1 in bit 11, row 11 by bit 10, row 0 by bit 9, and rows 1 through 9 by bits 8 through 0 of the byte), 16 central memory words per card. Literal input cards are read until a card identical to the previous 5/7/9 card (4/5/6/7/8/9 in column 2) is read. The next card can specify a new conversion mode.

Under NOS/BE, coded character data is exchanged with local batch card readers or card punches according to the translations shown in table A-2. As indicated in the table, additional card keypunch codes are available for input of the CDC characters ∨ and < or their ASCII equivalents ! and <. The 95-character set cannot be read or punched as coded character data.

Depending on an installation option, NOS/BE assumes an input deck has been punched either in 026 or in 029 keypunch code (regardless of the character set in use). The alternate keypunch codes can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or in columns 79 and 80 of any 7/8/9 card. The specified code translation remains in effect throughout the job unless it is reset by specification of the alternate code translation on a subsequent 7/8/9 card.

Under NOS/BE, a card with all of column 1 punched (that is, 12 punches in column 1) and all of one other column punched can be followed by 80-column cards of free-form binary data. These binary data cards are read or punched as described for NOS literal data until another card with 12 punches in column 1 and in one other column occurs, or until the job ends. The next card is interpreted as coded data.

Remote Batch Users

When card decks are read from remote batch devices, the ability to select alternate keypunch code translations depends upon the remote terminal equipment.

NOS Usage

Remote batch terminal line printer, punched card, and plotter character set support is described in the Remote Batch Facility (RBF) reference manual. RBF supports only character mode transmission to and from consoles through the network. Character mode is described under Network Access Method Terminal Transmission Code Sets in this appendix.

NOS/BE Usage

The remote batch terminal line printer, punched card, and plotter character set support is described in the INTERCOM reference manual.

Magnetic Tape Users

Coded character data to be copied from mass storage to magnetic tape is assumed to be represented in display code. NOS converts the data to external BCD code when writing a coded 7-track tape and to ASCII or EBCDIC code (as specified on the tape assignment statement) when writing a coded 9-track tape.

Because only 63 characters can be represented in 7-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code. Figure A-1 shows the differences in conversion that depend on which character set (63 or 64) the system uses. The ASCII character for the specified character code is shown in parentheses. The output arrow shows how the display code changes when it is written on tape in external BCD. The input arrow shows how the external BCD code changes when the tape is read and converted to display code.

63-Character Set

Display Code		External BCD		Display Code
00		16(%)		00
33(0)	Output $\longrightarrow$	12(0)	Input $\longrightarrow$	33(0)
63(:)		12(0)		33(0)

64-Character Set

Display Code		External BCD		Display Code
00(:)		12(0)		33(0)
33(0)	Output $\longrightarrow$	12(0)	Input $\longrightarrow$	33(0)
63(%)		16(%)		63(%)

Figure A-1. Magnetic Tape Code Conversions

Tables A-3 and A-4 show the character set conversions for 9-track tapes. Table A-3 lists the conversions to and from 7-bit ASCII character code and 6-bit display code. Table A-4 lists the conversions between 8-bit EBCDIC character code and 6-bit display code. Table A-5 shows the character set conversions between 6-bit external BCD and 6-bit display code for 7-track tapes.

If a lowercase ASCII or EBCDIC code is read from a 9-track coded tape, it is converted to its uppercase 6-bit display code equivalent. To read and write lowercase ASCII or EBCDIC characters, the user must assign the tape in binary mode and then convert the binary character data.

During binary character data transfers to or from 9-track magnetic tape, the 7-bit ASCII codes shown in table A-6 are read or written unchanged; the 8-bit hexadecimal EBCDIC codes shown in table A-7 also can be read or written unchanged. ASCII and EBCDIC codes cannot be read or written to 7-track magnetic tape as binary character data.

Two CDC utility products, FORM and the 8-Bit Subroutines, can be used to convert to and from EBCDIC data. Table A-7 contains the octal values of each EBCDIC code right-justified in a 12-bit byte with zero fill. This 12-bit EBCDIC code can also be produced using FORM and the 8-Bit Subroutines.

NETWORK ACCESS METHOD TERMINAL TRANSMISSION CODE SETS

There are two modes in which coded character data can be exchanged with a network terminal console. These two modes, character mode and transparent mode, correspond to the type of character code editing and translation performed by the network software during input and output operations. The transmission mode used by the network software for input can be selected by the terminal operator, using a Terminal Interface Program command (sometimes referred to as a terminal definition command). The transmission mode used by the network software for output can be selected by the application program providing the terminal facility service.

Character Mode Transmissions

Character mode is the initial and default mode used for both input and output transmissions. When the network software services the terminal in character mode, it translates input characters from the transmission code used by the terminal into the ASCII code shown in table A-6. The translation of a specific transmission code to a specific ASCII code depends on the terminal class the network software associates with the terminal. In character mode input, the parity of the terminal transmission code is not preserved in the corresponding ASCII code; the ASCII code received by the terminal-servicing facility program always has its eighth bit set to zero.

Character mode output is translated in a similar manner. The network software provides the parity bit setting appropriate for the terminal being serviced, even though translating from ASCII characters with zero parity bit settings.

The general case for code translations of character mode data is summarized in the following paragraphs. This generalized description permits use of only table A-6 to explain all specific cases. You can logically extend this generalized description to allow use of tables A-1 through A-5 as descriptions of character set mapping for various functions initiated from a terminal. Tables A-1 through A-5 are provided for your use while coding an application program to run under the operating system. They do not describe character transmissions between an application program and the network.

TABLE A-3. ASCII 9-TRACK CODED TAPE CONVERSION

ASCII				Display Code[†††]	
Code Conversion[†]		Character and Code Conversion[††]			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
20	space	00	NUL	space	55
21	!	7D	}	!	66
22	"	02	STX	"	64
23	#	03	ETX	#	60
24	$	04	EOT	$	53
25	%	05	ENQ	%	63
25	%	05	ENQ	space	55
26	&	06	ACK	&	67
27	'	07	BEL	'	70
28	(	08	BS	(	51
29	)	09	HT	)	52
2A	*	0A	LF	*	47
2B	+	0B	VT	+	45
2C	,	0C	FF	,	56
2D	-	0D	CR	-	46
2E	.	0E	SO	.	57
2F	/	0F	SI	/	50
30	0	10	DLE	0	33
31	1	11	DC1	1	34
32	2	12	DC2	2	35
33	3	13	DC3	3	36
34	4	14	DC4	4	37
35	5	15	NAK	5	40
36	6	16	SYN	6	41
37	7	17	ETB	7	42
38	8	18	CAN	8	43
39	9	19	EM	9	44
3A	:	1A	SUB	:	00
Display code 00 is undefined at sites using the 63-character set.					
3A	:	1A	SUB	:	63
3B	;	1B	ESC	;	77
3C	<	7B	{	<	72
3D	=	1D	GS	=	54
3E	>	1E	RS	>	73
3F	?	1F	US	?	71

ASCII				Display Code[†††]	
Code Conversion[†]		Character and Code Conversion[††]			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
40	@	60	`	@	74
41	A	61	a	A	01
42	B	62	b	B	02
43	C	63	c	C	03
44	D	64	d	D	04
45	E	65	e	E	05
46	F	66	f	F	06
47	G	67	g	G	07
48	H	68	h	H	10
49	I	69	i	I	11
4A	J	6A	j	J	12
4B	K	6B	k	K	13
4C	L	6C	l	L	14
4D	M	6D	m	M	15
4E	N	6E	n	N	16
4F	O	6F	o	O	17
50	P	70	p	P	20
51	Q	71	q	Q	21
52	R	72	r	R	22
53	S	73	s	S	23
54	T	74	t	T	24
55	U	75	u	U	25
56	V	76	v	V	26
57	W	77	w	W	27
58	X	78	x	X	30
59	Y	79	y	Y	31
5A	Z	7A	z	Z	32
5B	[	1C	FS	[	61
5C	\	7C	|	\	75
5D	[	01	SOH	]	62
5E	^	7E	~	^	76
5F	_	7F	DEL	_	65

[†]When these characters are copied from or to a tape, the characters remain the same and the code changes from/to ASCII to/from display code.

[††]These characters do not exist in display code. When the characters are copied from a tape, each ASCII character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 61_{16}, from tape, it writes an uppercase A, 01_8.

[†††]A display code space always translates to an ASCII space.

Table A-6 contains the ASCII 128-character set supported by the Network Access Method. A 96-character subset consists of the rightmost six columns and includes the 95-character graphic subset referenced previously in this appendix; the deletion character (DEL) is not a graphic character. A 64-character subset consists of the middle four columns. Note that 6-bit display code equivalents exist for the characters in this 64-character subset only.

Although the network supports the 128-character set, some terminals restrict output to a smaller subset. This restriction is supported by replacing the control characters in columns 0 and 1 of table A-6 with blanks to produce the 96-character subset, and, additionally, replacing the characters in columns 6 and 7 with the corresponding characters from columns 4 and 5, respectively, to produce the 64-character subset.

TABLE A-4. EBCDIC 9-TRACK CODED TAPE CONVERSION

EBCDIC				Display Code†††	
Code Conversion†		Character and Code Conversion††			
Code (Hex)	Char	Code (Hex)	Char	ASCII Char	Code (Octal)
40	space	00	NUL	space	55
4A	¢	1C	IFS	[	61
4B	.	0E	SO	.	57
4C	<	C0	{	<	72
4D	(	16	BS	(	51
4E	+	0B	VT	+	45
4F	|	D0	}	!	66
50	&	2E	ACK	&	67
5A	!	01	SOH	]	62
5B	$	37	EOT	$	53
5C	*	25	LF	*	47
5D	)	05	HT	)	52
5E	;	27	ESC	;	77
5F	¬	A1	~	/	76
60	-	0D	CR	-	46
61	'	0F	SI	'	50
6B	,	0C	FF	,	56
6C	%	2D	ENQ	%	63
6C	%	2D	ENQ	space	55
6D	_	07	DEL	_	65
6E	>	1E	IRS	>	73
6F	?	1F	IUS	?	71
7A	:	3F	SUB	:	00
Display code 00 is undefined at sites using the 63-character set.					
7A	:	3F	SUB	:	63
7B	#	03	ETX	#	60
7C	@	79	\	@	74
7D	'	2F	BEL	'	70
7E	=	1D	IGS	=	54
7F	"	02	STX	"	64
C1	A	81	a	A	01
C2	B	82	b	B	02
C3	C	83	c	C	03
C4	D	84	d	D	04
C5	E	85	e	E	05
C6	F	86	f	F	06
C7	G	87	g	G	07
C8	H	88	h	H	10
C9	I	89	i	I	11
D1	J	91	j	J	12
D2	K	92	k	K	13
D3	L	93	l	L	14
D4	M	94	m	M	15
D5	N	95	n	N	16
D6	O	96	o	O	17
D7	P	97	p	P	20
D8	Q	98	q	Q	21
D9	R	99	r	R	22
E0	\	6A	|	\	75
E2	S	A2	s	S	23
E3	T	A3	t	T	24
E4	U	A4	u	U	25
E5	V	A5	v	V	26
E6	W	A6	w	W	27
E7	X	A7	x	X	30
E8	Y	A8	y	Y	31
E9	Z	A9	z	Z	32
F0	0	10	DLE	0	33
F1	1	11	DC1	1	34
F2	2	12	DC2	2	35
F3	3	13	TM	3	36
F4	4	3C	DC4	4	37
F5	5	3D	NAK	5	40
F6	6	32	SYN	6	41
F7	7	26	ETB	7	42
F8	8	18	CAN	8	43
F9	9	19	EM	9	44

†All EBCDIC codes not listed translate to display code 55_8 (space). A display code space always translates to an EBCDIC space.

††These characters do not exist in display code. When the characters are copied from a tape, each EBCDIC character is changed to an alternate display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 81_{16}, from tape, it writes an uppercase A, 01_8.

†††When these characters are copied from or to a tape, the characters remain the same (except EBCDIC codes $4A_{16}$, $4F_{16}$, $5A_{16}$, and $5F_{16}$) and the code changes from/to EBCDIC to/from display code.

Similarly, input from a device may be limited to a smaller subset by the device itself because the device cannot produce the full 128-character set. A character input from a device using a character set other than ASCII is converted to an equivalent ASCII character; characters without ASCII character equivalents are replaced by the ASCII blank character.

An application can also cause character replacement (as described previously for output) as well as character conversion, by requesting display-coded input from the network.

The 7-bit hexadecimal code value for each character consists of the character's column number in the table, followed by its row number. For example, N is in row E of column 4, so its value is $4E_{16}$.

TABLE A-5. 7-TRACK CODED TAPE CONVERSIONS

External BCD	ASCII Character	Octal Display Code	External BCD	ASCII Character	Octal Display Code
01	1	34	40	-	46
02	2	35	41	J	12
03	3	36	42	K	13
04	4	37	43	L	14
05	5	40	44	M	15
06	6	41	45	N	16
07	7	42	46	O	17
10	8	43	47	P	20
11	9	44	50	Q	21
12†	0	33	51	R	22
13	=	54	52	!	66
14	"	64	53	$	53
15	@	74	54	*	47
16†	%	63	55	'	70
17	[	61	56	?	71
20	space	55	57	>	73
21	/	50	60	+	45
22	S	23	61	A	01
23	T	24	62	B	02
24	U	25	63	C	03
25	V	26	64	D	04
26	W	27	65	E	05
27	X	30	66	F	06
30	Y	31	67	G	07
31	Z	32	70	H	10
32	]	62	71	I	11
33	,	56	72	<	72
34	(	51	73	.	57
35		65	74	)	52
36	#	60	75	\	75
37	&	67	76	^	76
			77	;	77

†As explained in the text of this appendix, conversion of these codes depends on whether the tape is being read or written.

Transparent Mode Transmissions

Transparent mode is selected separately for input and output transmissions. During transparent mode input, the parity bit is stripped from each terminal transmission code (unless the N parity option has been selected by a Terminal Interface Program command), and the transmission code is placed in an 8-bit byte without translation to 7-bit ASCII code. Line transmission protocol characters are deleted from a mode 4C terminal input stream.

When the 8-bit bytes arrive in the host computer, a terminal servicing facility program such as the Interactive Facility can right-justify the bytes within a 12-bit byte. Upon transmission of 12-bit bytes from the host computer, the leftmost 4 bits (bits 11 through 8) are discarded.

During transparent mode output, processing similar to that performed for input occurs. The code in each 8-bit byte received by the network software from the terminal servicing facility program is not translated. The parity bit appropriate for the terminal class being used is altered as indicated by the parity option in effect for the terminal. The codes are then output in transmission bytes appropriate for the codes associated with the terminal class being used. Line transmission protocol characters are inserted into a mode 4C terminal output stream.

TABLE A-6. FULL ASCII CHARACTER SET

← 128-Character Set →

← 96-Character Subset →

← 64-Character Subset →

					b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits	b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL 000	DLE 020	SP 040	0 060	@ 100	P 120	` 140	p 160
	0	0	0	1	1	SOH 001	DC1 021	! 041	1 061	A 101	Q 121	a 141	q 161
	0	0	1	0	2	STX 002	DC2 022	" 042	2 062	B 102	R 122	b 142	r 162
	0	0	1	1	3	ETX 003	DC3 023	# 043	3 063	C 103	S 123	c 143	s 163
	0	1	0	0	4	EOT 004	DC4 024	$ 044	4 064	D 104	T 124	d 144	t 164
	0	1	0	1	5	ENQ 005	NAK 025	% 045	5 065	E 105	U 125	e 145	u 165
	0	1	1	0	6	ACK 006	SYN 026	& 046	6 066	F 106	V 126	f 146	v 166
	0	1	1	1	7	BEL 007	ETB 027	' 047	7 067	G 107	W 127	g 147	w 167
	1	0	0	0	8	BS 010	CAN 030	(050	8 070	H 110	X 130	h 150	x 170
	1	0	0	1	9	HT 011	EM 031	) 051	9 071	I 111	Y 131	i 151	y 171
	1	0	1	0	A	LF 012	SUB 032	* 052	: 072	J 112	Z 132	j 152	z 172
	1	0	1	1	B	VT 013	ESC 033	+ 053	; 073	K 113	[133	k 153	{ 173
	1	1	0	0	C	FF 014	FS 034	, 054	< 074	L 114	\ 134	l 154	| 174
	1	1	0	1	D	CR 015	GS 035	- 055	= 075	M 115	] 135	m 155	} 175
	1	1	1	0	E	SO 016	RS 036	. 056	> 076	N 116	^ 136	n 156	~ 176
	1	1	1	1	F	SI 017	US 037	/ 057	? 077	O 117	_ 137	o 157	DEL 177

LEGEND:

Numbers under characters are the octal values for the 7-bit character codes used within the network.

TABLE A-7. FULL EBCDIC CHARACTER SET

Hexa-decimal EBCDIC Code	Octal 12-Bit EBCDIC Code	EBCDIC Graphic Character†	EBCDIC Control Character	Hexa-decimal EBCDIC Code	Octal 12-Bit EBCDIC Code	EBCDIC Graphic Character†	EBCDIC Control Character
00	0000		NUL	41	0101		undefined
01	0001		SOH	thru	thru		
02	0002		STX	49	0111		
03	0003		ETX	4A	0112	¢	
04	0004		PF	4B	0113	.	
05	0005		HT	4C	0114	<	
06	0006		LC	4D	0115	(	
07	0007		DEL	4E	0116	+	
08	0010		undefined	4F	0117	|	
09	0011		undefined	50	0120	&	
0A	0012		SMM	51	0121		undefined
0B	0013		VT	thru	thru		
0C	0014		FF	59	0131		
0D	0015		CR	5A	0132	!	
0E	0016		SO	5B	0133	$	
0F	0017		SI	5C	0134	*	
10	0020		DLE	5D	0135	)	
11	0021		DC1	5E	0136	;	
12	0022		DC2	5F	0137	¬	
13	0023		TM	60	0140	-	
14	0024		RES	61	0141	/	
15	0025		NL	62	0142		undefined
16	0026		BS	thru	thru		
17	0027		IL	69	0151		
18	0030		CAN	6A	0152	¦	
19	0031		EM	6B	0153	,	
1A	0032		CC	6C	0154	%	
1B	0033		CU1	6D	0155	_	
1C	0034		IFS	6E	0156	>	
1D	0035		IGS	6F	0157	?	
1E	0036		IRS	70	0160		undefined
1F	0037		IUS	thru	thru		
20	0040		DS	78	0170		
21	0041		SOS	79	0171	`	
22	0042		FS	7A	0172	:	
23	0043		undefined	7B	0173	#	
24	0044		BYP	7C	0174	@	
25	0045		LF	7D	0175	'	
26	0046		ETB or EOB	7E	0176	=	
27	0047		ESC or PRE	7F	0177	"	
28	0050		undefined	80	0200		undefined
29	0051		undefined	81	0201	a	
2A	0052		SM	82	0202	b	
2B	0053		CU2	83	0203	c	
2C	0054		undefined	84	0204	d	
2D	0055		ENQ	85	0205	e	
2E	0056		ACK	86	0206	f	
2F	0057		BEL	87	0207	g	
30	0060		undefined	88	0210	h	
31	0061		undefined	89	0211	i	
32	0062		SYN	8A	0212		undefined
33	0063		undefined	thru	thru		
34	0064		PN	90	0220		
35	0065		RS	91	0221	j	
36	0066		UC	92	0222	k	
37	0067		EOT	93	0223	l	
38	0070		undefined	94	0224	m	
39	0071		undefined	95	0225	n	
3A	0072		undefined	96	0226	o	
3B	0073		CU3	97	0227	p	
3C	0074		DC4	98	0230	q	
3D	0075		NAK	99	0231	r	
3E	0076		undefined	9A	0232		undefined
3F	0077		SUB	thru	thru		
40	0100	space		A0	0240		

TABLE A-7. FULL EBCDIC CHARACTER SET (Contd)

Hexa-decimal EBCDIC Code	Octal 12-Bit EBCDIC Code	EBCDIC Graphic Character[†]	EBCDIC Control Character	Hexa-decimal EBCDIC Code	Octal 12-Bit EBCDIC Code	EBCDIC Graphic Character[†]	EBCDIC Control Character
A1	0241	~		D7	0327	P	
A2	0242	s		D8	0330	Q	
A3	0243	t		D9	0331	R	
A4	0244	u		DA	0332		undefined
A5	0245	v		thru	thru		
A6	0246	w		DF	0337		
A7	0247	x		E0	0340	\	
A8	0250	y		E1	0341		undefined
A9	0251	z		E2	0342	S	
AA	0252		undefined	E3	0343	T	
thru	thru			E4	0344	U	
BF	0277			E5	0345	V	
C0	0300	{		E6	0346	W	
C1	0301	A		E7	0347	X	
C2	0302	B		E8	0350	Y	
C3	0303	C		E9	0351	Z	
C4	0304	D		EA	0352		undefined
C5	0305	E		EB	0353		undefined
C6	0306	F		EC	0354	⑁	
C7	0307	G		ED	0355		undefined
C8	0310	H		thru	thru		
C9	0311	I		EF	0357		
CA	0312		undefined	F0	0360	0	
CB	0313		undefined	F1	0361	1	
CC	0314	⑀		F2	0362	2	
CD	0315	⑂	undefined	F3	0363	3	
CE	0316			F4	0364	4	
CF	0317		undefined	F5	0365	5	
D0	0320	}		F6	0366	6	
D1	0321	J		F7	0367	7	
D2	0322	K		F8	0370	8	
D3	0323	L		F9	0371	9	
D4	0324	M		FA	0372	|	
D5	0325	N		FB	0373		undefined
D6	0326	O		thru	thru		
				FF	0377		

[†]Graphic characters shown are those used on the IBM System/370 standard (PN) print train. Other devices support subsets or variations of this character graphic set.

DIAGNOSTICS

Diagnostic messages that can be issued by Sort/Merge are listed in table B-1. If an error occurs, the error level and the error number are written on the error file or the listing file. The diagnostic message is written on the next line. For errors numbered 1 through 40, the parameter statement that is in error is written on another line, and a pointer indicates the position of the error in the parameter statement.

An error can be one of the following levels:

T Trivial; a trivial diagnostic results from a usage that is syntactically correct but questionable.

W Warning; a warning diagnostic results from a usage that is syntactically incorrect, but from which Sort/Merge has been able to recover by making an assumption about what was intended.

F Fatal; a fatal diagnostic results when Sort/Merge cannot resolve a syntactic or semantic error. The remaining parameter statements are read, but no Sort/Merge processing occurs.

C Catastrophic; a catastrophic error causes immediate Sort/Merge termination.

Specifying the STATUS parameter or calling the SM5ST procedure prevents job step termination; the status variable indicates the highest level of error that occurred.

TABLE B-1. SORT/MERGE DIAGNOSTICS

Error Number	Error Level	Message	Significance	Action
1	F	A SPACE OR , EXPECTED	A space or a comma must separate parameters in a control statement or directive file. A space or a comma must also separate values in a set of values specified for a parameter.	Insert a space or a comma between each parameter and between values specified for a parameter.
2	C	CHARACTERS USED OUTSIDE A STRING OR NAME MUST BE LETTER, DIGIT) = (′ , OR SPACE	Unless in a literal string or in the name of a user-supplied entity, only alphabetic and numeric characters and the special characters) = (′ , and space are valid in a control statement or directive file.	Replace invalid characters with valid characters.
3	F	A VALUE SHOULD FOLLOW THIS KEYWORD	At least one value must be specified following a parameter keyword.	Insert a value following the parameter keyword.
4	C	UNDEFINED COLLATING SEQUENCE	The name of a key type in a KEY parameter is not a predefined collating sequence or a sequence you have defined with SEQx parameters.	Change the name of the key type to one of the predefined collating sequences or a sequence you have defined.
5	F	CHARACTER AFTER KEYWORD MUST BE SPACE OR =	A parameter keyword must be separated from its value by an = or space.	Insert an = after the parameter keyword.
6	F	INVALID KEYWORD DETECTED	The indicated keyword is not a valid procedure call keyword.	Determine valid procedure call keywords.
7	F	UNLESS IN A STRING, = SHOULD FOLLOW ONLY A KEYWORD	Unless an = is in a literal string, = is only valid immediately following a parameter keyword.	Replace the invalid = with a valid character.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
9	F	( ) MUST ENCLOSE A VALUE	Parentheses in a control statement or directive file must enclose a parameter value.	Place a parameter value between the parentheses.
10	F	CHARACTER AFTER = MUST BE LETTER, DIGIT ($ ′ OR SPACE	Only an alphabetic or numeric character or one of the special characters ($ ′ or space can follow an =.	Replace the character following the = with a valid character.
11	F	THIS VALUE SHOULD BE A STRING (DELIMITED WITH ′S)	The indicated value must be a literal string enclosed in apostrophes.	Enclose the indicated value in apostrophes.
13	F	THIS COMMA SHOULD BE FOLLOWED BY A LETTER, DIGIT, SPACE , OR (	The indicated comma must be followed by an alphabetic or numeric character or one of the special characters , (or space.	Replace the character following the indicated comma with a valid character.
14	F	THIS CHARACTER MUST BE A LETTER, DIGIT (′ SPACE OR ,	The indicated character must be an alphabetic or numeric character or one of the special characters (′ , or space.	Replace the indicated character with a valid character.
15	F	TOO MANY PARAMETER POSITIONS HAVE BEEN SPECIFIED	When specifying parameters by position rather than by keyword, a maximum of 31 positions is allowed; you have probably included too many commas.	Check your control statement for excess commas.
17	F	CHARACTER AFTER) MUST BE SPACE) OR ,	A right parenthesis can only be followed by another right parenthesis, a comma, or a space.	Replace the invalid character after the right parenthesis with a valid character.
18	F	CHARACTER AFTER (MUST BE A LETTER, DIGIT $ ′ (OR SPACE	A left parenthesis can only be followed by an alphabetic or numeric character, another left parenthesis, a $ ′ or space.	Replace the invalid character after the left parenthesis with a valid character.
19	F	THE HIGH AND LOW VALUES OF THIS RANGE MUST BE AN INTEGER, R1, R2, R3, R1G, EF OR EFG	If a range of values is specified for the KEY, SUM, or ENR parameter, the high and low values of the range must be integers or the CCL variables R1, R2, R3, R1G, EF, or EFG.	Replace the invalid range value with an integer, R1, R2, R3, R1G, EF, or EFG.
20	F	THE HIGH AND LOW VALUES OF THIS RANGE MUST BE A STRING	If the SEQS parameter specifies a range of values, the high and low values of the range must be literal strings enclosed in apostrophes.	Replace the invalid range with a literal string enclosed in apostrophes.
21	F	CHARACTER FOLLOWING AN INTEGER MUST BE) .. , SPACE OR LETTER	Only a) .. (indicating a range of values), a comma, a space, or a letter can follow an integer.	Replace the invalid character following the integer with a valid character.
22	F	POSITIONAL VALUES ARE NOT ALLOWED IN A VALUE-SET	In a KEY or SUM parameter value-set, key or sum field characteristics must be specified in the order shown in the parameter formats; values cannot be omitted and replaced by commas.	In a KEY parameter, specify field position, length and type if you specify order; specify field position and length if you specify type. In a SUM parameter, specify field position, length and type if you specify repetitions.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
23	F	A PERIOD CANNOT BE USED IN A PARAMETER-LIST	A period is not a valid separator in a parameter list.	Replace the invalid period with a valid separator; refer to the parameter syntax to determine valid separators.
24	F	A RANGE IS ONLY ALLOWED AS A VALUE FOR ENR, KEY OR SUM PARAMETERS	A range of values can be specified only for the ENR, KEY, or SUM parameters; single values must be specified for other parameters.	Replace the invalid range with a single value.
25	F	UNLESS $CHAR(N) IS USED, ONLY ONE SET OF NESTED PARENTHESES IS ALLOWED	More than two parentheses can appear together only when $CHAR(n) is used in a SEQS parameter to specify steps in a collating sequence.	Delete the extra parentheses.
26	F	UNLESS WITHIN A STRING, NEITHER . NOR = IS ALLOWED WITHIN A VALUE-SET	A set of values specified for a parameter can contain a period or an = only in a literal string.	Correct the syntax for the indicated values.
27	F	THIS VALUE SHOULD BE A FILE OR PROC NAME	The indicated value must be the name of a file or procedure.	Use a file or procedure name for the indicated value.
29	F	PARENTHESES MUST BE MATCHED IN THE PARAMETER-LIST	If a right or left parenthesis appears in a parameter list, it must be matched in the list by its opposite.	Insert the missing parenthesis.
30	F	TOO FEW SETS GIVEN FOR THE KEYWORD ENDED HERE	The indicated keyword must have a specific set of values specified for it; a value or values are missing.	Insert the values that must be specified for the indicated keyword.
31	F	TOO MANY VALUE-SETS GIVEN FOR THIS KEYWORD-VALUE-LIST	The indicated keyword must have a specific set of values specified for it; too many values have been specified.	Determine the values that can be specified for the indicated keyword.
32	F	THIS VALUE SHOULD BE AN INTEGER	The indicated value must be an integer.	Replace the invalid value with an integer value.
33	F	TOO MANY VALUES SPECIFIED IN THIS VALUE-SET	Too many values have been specified for the indicated set of values in a list of parameter values.	Determine the values that can be specified in the indicated set of values.
34	F	TOO FEW VALUES SPECIFIED IN THIS VALUE-SET	Too few values have been specified for the indicated set of values in a list of parameter values.	Determine the values that can be specified in the indicated set of values.
35	F	ONLY A ,) .. OR SPACE CAN FOLLOW A PARAMETER VALUE	Parameter values must be followed by a ,) .. or space.	Replace the invalid character following the parameter value with a valid character.
37	F	THIS PARAMETER VALUE MUST BE A SET OR A RANGE	The indicated parameter value cannot be a single value; the value must be a set of values or a range of values.	Replace the indicated parameter value with a valid set or range of values.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
50	W	DIALOG PARAMETER CAN ONLY BE SPECIFIED IN A CONTROL STATEMENT	The DIALOG parameter cannot be in a directive file.	Specify the DIALOG parameter in a control statement.
51	F	ONLY 3 VALUES CAN BE GIVEN IN KEY/SUM VALUE-SET IF F..L FORMAT IS USED	Values in a KEY or SUM parameter value-set are separated by commas or spaces. If you use the first..last form to specify key or sum field position and length, a maximum of two other values (separated by commas or spaces) is allowed.	Correct the KEY or SUM parameter so that no more than three values are specified.
52	F	FILE, PROC AND COL-SEQ NAMES MUST BEGIN WITH A LETTER	A letter (A-Z) must be the first character in a file name, procedure name, or the name of a collating sequence you have defined.	Begin each file name, procedure name, and collating sequence name with a letter.
53	F	VALUES ALLOWED FOR THE FASTIO PARAMETER ARE YES AND NO	Valid values for the FASTIO parameter are YES or NO, which can be specified as Y or N.	Specify a valid value for the FASTIO parameter.
54	W	MORE THAN 100 INPUT FILES GIVEN, ONLY THE FIRST 100 SORTED	No more than 100 files can be sorted in any one job step; files specified in excess of 100 are ignored.	The files in excess of 100 can be sorted in a separate job step.
56	F	IF THE FIRST VALUE OF A SEQS IS A RANGE THEN ALL OTHER VALUES OF THAT SEQS MUST BE RANGES OF THE SAME LENGTH	Single characters cannot follow a range of characters in a SEQS parameter.	Correct the SEQS parameter so that it consists of ranges of characters with the same number of characters in each range.
57	F	WHEN LOW..HIGH IS USED AS A VALUE FOR SEQS LOW MUST BE LESS THAN HIGH	If a range of characters is used in a SEQS parameter, the first character in the range must be of less value than the last.	Correct the SEQS parameter so that the first character in the range is less than the last.
58	F	ONLY SINGLE CHARACTER STRINGS OR $CHAR CAN BE USED AS VALUES FOR SEQS	Each character specified in a SEQS parameter must be enclosed in apostrophes and separated from other characters by .. or a comma; characters cannot be grouped together. $CHAR(n) must specify a single character.	Correct the SEQS parameter.
59	F	FOR $CHAR(N), N MUST BE A POSITIVE DECIMAL INTEGER	If $CHAR(n) is used in a SEQS parameter to specify a value step consisting of one character, n must be a positive decimal integer that corresponds to the number of the intended character in your character set.	Specify n as a positive decimal integer.
60	F	CONTRADICTORY VALUES FOR LO PARAMETER SHOULD NOT BE SPECIFIED	You cannot specify LO=OFF along with S and/or A.	Correct the LO parameter.
61	F	FILE NAMES MUST NOT EXCEED 7 CHARACTERS IN LENGTH	A file name can be no longer than 7 characters.	Change the name of the file to a name that is 7 characters or fewer in length.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
62	C	THE MAXIMUM LENGTH OF A LINE IS 240 CHARACTERS	The SORT5 or MERGE control statement and its associated parameter list can be no longer than 240 characters (periods used as continuation indicators are not counted).	Parameters can be placed in a directive file so that the control statement is 240 characters or less.
63	F	VALUES ALLOWED FOR THE DIALOG PARAMETER ARE YES AND NO	Valid values for the DIALOG parameter are YES or NO, which can be specified as Y or N.	Specify a valid value for the DIALOG parameter.
64	F	A SPACE OR , MUST FOLLOW THE SORT OR MERGE VERBS IN DIR. FILES	In a directive file, any statement that begins with SORT or with MERGE must have a space or a comma immediately following SORT or MERGE.	Insert a space or a comma immediately following SORT or MERGE in your directive file.
65	C	A PERIOD MUST FOLLOW SORT5 OR MERGE	In a control statement, the word SORT5 or the word MERGE must be immediately followed by a period.	Insert a period immediately following the word SORT5 or the word MERGE.
67	F	ALL DIRECTIVE FILE NAMES MUST BE UNIQUE	A directive file cannot have the same name as any other directive file used in the sort or merge.	Give all directive files different names.
68	F	LINES CONTINUING THE SORT5 OR MERGE PARAMETER-LIST MUST BEGIN WITH A PERIOD	Any line of a SORT5 or a MERGE control statement that is continued beyond the first line must begin with one period.	Begin each continuation line with one period.
69	F	FILE NAMES CAN CONTAIN ONLY LETTERS AND DIGITS	A file name cannot contain any special characters.	Omit special characters from the file name.
70	F	LENGTH MUST BE SPECIFIED IF TYPE IS SPECIFIED	If the key type is other than the default ASCII6, or if ASCII6 is explicitly specified, key length cannot be omitted.	Specify key length.
71	F	KEY LENGTH FOR KEY OF KEYTYPE REAL MUST BE 10	Keys declared to be of type REAL must occupy a full computer word and must be specified as 10 bytes in length.	Specify length of REAL key as 10.
72	F	ONLY THE VALUES A OR D ARE ALLOWED FOR KEY ORDER	Sort order can be specified as ascending with A or as descending with D.	Specify either A or D for sort order; or omit specification of sort order and ascending is assumed.
73	W	UNEQUAL FL FOR INPUT AND OUTPUT FILES WITH RT=F OR RT=Z	If you specify different fixed length for input and output files, Sort/Merge compresses or expands records to fit into the fixed length.	For the most efficient processing, specify the same FL for input and output files.
74	F	VALUES ALLOWED FOR THE EL PARAMETER ARE T, W, F OR C	The only values that can be specified for the EL parameter are T, W, F, or C.	Specify T, W, F, or C as the value of the EL parameter.
75	F	VALUES ALLOWED FOR THE LO PARAMETER ARE OFF, S AND A	The only values that can be specified for the LO parameter are OFF, S, or A.	Specify OFF, S, or A as the value of the LO parameter.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
76	F	VALUES ALLOWED AS EXPRESSIONS ARE INTEGER, R1, R2, R3, R1G, EF OR EFG	The only values that can be specified as expressions are integers or the CCL variables R1, R2, R3, R1G, EF, or EFG.	Specify a valid expression.
77	F	$NULL SHOULD NOT BE USED AS A VALUE WITHIN A SET OF VALUES	$NULL is a reserved file name signifying a null file; it must not be specified in a list of multiple files. $NULL can be specified as a single value for a parameter.	Delete $NULL from the list of files, or specify $NULL as the single parameter value.
78	F	VALUES ALLOWED FOR THE ST PARAMETER ARE R1, R2, R3, R1G, EF OR EFG	The only values that can be specified for the STATUS parameter are the CCL variables R1, R2, R3, R1G, EF, or EFG.	Specify a valid value for the STATUS parameter.
79	F	VALUES ALLOWED FOR THE RETAIN PARAMETER ARE YES AND NO	The only values that can be specified for the RETAIN parameter are YES or NO, which can be abbreviated as Y or N.	Specify a valid value for the RETAIN parameter.
80	F	OWNMRL AND OWNFL SHOULD NOT BOTH BE SPECIFIED FOR THE SAME SORT	In a sort with owncode routines, record length must be specified with either the OWNMRL parameter or the OWNFL parameter.	Specify record length with the OWNFL parameter for F or Z type records; specify record length with the OWNMRL parameter for all other record types.
81	F	SUM AND KEY FIELDS MAY NOT OVERLAP	The same characters in a record cannot be included in a sum field and a key field.	Correct SUM parameter and/ or KEY parameter so that the same characters are not included in both field descriptions.
82	C	LINES IN A DIRECTIVE FILE MUST BE NO LONGER THAN 100 CHARACTERS IN LENGTH	A directive file statement, beginning with SORT or MERGE, can be 240 characters in length, but each line in the statement must contain no more than 100 characters.	Use continuation lines in your directive file so that each line contains no more than 100 characters.
83	F	REP. VALUE OF A SUM FIELD MUST BE AN INTEGER	In a SUM parameter, the number of fields to be summed must be expressed as an integer value.	Specify number of fields to be summed as an integer.
84	F	VALUES ALLOWED FOR OWNT PARAMETER MUST BE OLD OR NEW	The only values that can be specified for the OWNT parameter are OLD or NEW.	Specify OLD or NEW as the value for the OWNT parameter.
85	F	IN A KEY/SUM VALUE-SET USING FIRST..LAST, FIRST MUST BE LESS THAN LAST	In a KEY or SUM parameter, if field length and position are specified as the first and last bytes or bits of the field then the first byte or bit must be less than the last byte or bit (the leftmost byte or bit in a record is counted as number 1).	Correct the KEY or SUM parameter.
86	F	INVALID DATA-TYPE FOR SUM FIELD SPECIFIED	In a SUM parameter, sum field type must be the name of a numeric data format except REAL; type cannot be a collating sequence.	Correct the SUM parameter.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
87	C	LINES OF A DIR FILE MUST BEGIN WITH SORT OR MERGE VERB, OR BE A CONTINUATION	In a directive file, each line must begin with the word SORT or MERGE or, if the line is a continuation line, with a character of your parameter list or one or more spaces.	Correct the directive file; determine the proper format for directive files.
88	F	BYTE/BIT POSITION OR LENGTH OF A SUM/KEY FIELD MUST BE AN INTEGER	In a SUM or KEY parameter, field length and position must be specified as integers, counting the leftmost byte or bit in a record as number 1.	Correct the SUM or KEY parameter. (Make sure two sets of parentheses enclose the value-set.)
89	F	SEQN VALUE MUST NOT BE ONE OF THE PREDEFINED KEY-TYPES	When you are defining your own collating sequence with SEQx parameters, the name of your collating sequence cannot be the same as the name of one of the predefined collating sequences.	Give your collating sequence a unique name that is not the same as one of the predefined collating sequences.
90	F	SEQR AND SEQA MAY BE SPECIFIED ONLY ONCE FOR EACH COLLATING SEQUENCE	When you are defining your own collating sequence with SEQx parameters, SEQR and SEQA must not be specified more than once.	Specify SEQR and SEQA only once.
91	F	SEQN MUST BE SPECIFIED BEFORE ANY OTHER SEQ PARAMETER CAN BE GIVEN	When you are defining your own collating sequence with SEQx parameters, the SEQN parameter naming your collating sequence must be the first SEQx parameter.	Specify SEQN before you specify any other SEQx parameter.
92	F	VALUES ALLOWED FOR SEQR AND SEQA ARE YES AND NO	The only values that can be specified for the SEQR and SEQA parameters are YES or NO, which can be abbreviated as Y or N.	Specify a valid value for the SEQR and SEQA parameters.
93	F	SEQN SHOULD NOT BE GIVEN WITHOUT GIVING SEQS OR SEQR BEFORE THE NEXT SEQN	When you are defining your own collating sequence with SEQx parameters, you must specify the characters in your collating sequence with SEQS and SEQR parameters before naming another collating sequence.	Correct the order of the SEQx parameters.
94	F	VALUES ALLOWED IN SEQS PARAMETER ARE STRINGS AND $CHAR(N)	When you are defining your own collating sequence with SEQx parameters, the only valid values for SEQS parameters are characters enclosed in apostrophes or $CHAR(n) where n is the number of the character in your character set.	Correct the SEQS parameters.
95	F	TWO RANGES OF CHARACTERS USED IN THE SAME SEQS MUST BE THE SAME LENGTH	When you are defining your own collating sequence with SEQx parameters, the ranges in any one SEQS parameter must all be the same length.	Correct the SEQS parameter so that it specifies ranges that are all the same length.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
96	F	VALUE GIVEN IN THE $CHAR FUNCTION MUST BE AN INTEGER	When you are defining your own collating sequence with SEQx parameters and are using $CHAR(n) to specify the characters in the sequence, n must be a decimal integer corresponding to the number of the intended character in your character set.	Specify n of $CHAR(n) as a decimal integer.
97	F	CHARACTERS GIVEN IN A SINGLE COLLATING SEQUENCE MUST APPEAR ONLY ONCE	When you are defining your own collating sequence with SEQx parameters, a character cannot appear more than once in all the SEQS parameters.	Specify each character only once in a collating sequence.
98	W	COL-SEQUENCE NAMES LONGER THAN 10 CHARACTERS ARE TRUNCATED	You can specify a collating sequence name of up to 31 characters in length, but only the first 10 characters are significant.	None.
99	F	NUMBER OF KEY AND SUM CHARACTERS MUST BE NO MORE THAN 255	A record cannot contain more than 255 characters that are defined as key or sum fields.	Specify the total number of key and sum characters together as 255 or less.
100	C	SIGNED NUMERIC CHARACTER FIELD IS TOO SHORT	A key field defined by a KEY parameter or SM5KEY call as one of the signed numeric data formats is not long enough to contain the data.	Redefine the key field as long enough to contain all the numeric characters and the sign.
101	F	NO PARAMETER CAN BE SPECIFIED MORE THAN ONCE ON A LINE	Parameters that can be specified more than once must be on separate lines.	Inspect parameters and correct error(s).
102	F	DUPLICATE PARAMETER SPECIFIED	Some parameters can only be specified once.	Inspect parameters and correct error(s).
103	C	EITHER AN INPUT FILE, OWN1 OR OWN2 MUST BE GIVEN	No sort input file given.	Specify a sort input file, or OWN1 or OWN2.
104	C	EITHER AN OUTPUT FILE, OWN3 OR OWN4 MUST BE GIVEN	No sort output file given.	Specify a sort output file, or OWN1 or OWN2.
107	F	INCORRECT PARAMETER VALUE	A value specified as a procedure call parameter is not a valid value.	Check values that can be specified for the procedure call parameter.
108	F	TOO FEW/MANY PARAMETERS	Either too few or too many parameters have been specified in a procedure call.	Determine parameters that can be specified in the procedure call.
110	F	CALL OUT OF SEQUENCE	A call to a Sort/Merge FORTRAN procedure is not in the proper order.	Determine procedure call order.
111	F	INVALID NAME GIVEN FOR SM5SEQN	The name of a user-defined collating sequence specified with the SM5SEQN call must begin with a letter and consist of letters, digits, and the special characters ? # @ _.	Correct the name specified with the SM5SEQN call; enclose the name in apostrophes.

TABLE B-1. SORT/MERGE DIAGNOSTICS (Contd)

Error Number	Error Level	Message	Significance	Action
112	F	ONLY SINGLE CHARACTER LITERALS CAN BE USED AS PARAMETERS OF SM5SEQS	Each character in a user-defined collating sequence specified as parameters of the SM5SEQS procedure must be specified separately, enclosed in apostrophes, and separated from other characters by commas.	Specify each character in your collating sequence as a separate literal enclosed in apostrophes.
113	A	RECORD LENGTH IS GREATER THAN MAXIMUM RL - 5000	Record length is set by the full length (FL) or maximum record length (MRL) file information table (FIT). Also, the OWNFL and OWNMRL parameters or SM5OFL and SM5OMRL procedures specify the length of a record. Record length cannot exceed 5000 characters.	Create a new file with record length of 5000 characters or less.
114	F	SUM FIELDS MAY NOT BE GREATER THAN 60 BITS	Summing is handled with integer arithmetic. Integer and binary sum fields can be no more than 60 bits long.	Correct the specification of the sum field.
115	C	RETAIN AND SUMMING MAY NOT BE SPECIFIED AT THE SAME TIME	You cannot specify both the RETAIN and the SUM parameter in the same sort or merge.	Remove the RETAIN or SUM parameter.
116	W	MRL/FL IS NOT LARGE ENOUGH TO ACCOMMODATE THE SPECIFIED KEYS	The record size (MRL or FL) is not large enough to accommodate the specified keys.	Change your key or file description.
117	W	FASTIO PARAMETER DISABLED FOR FILES NOT ON MASS STORAGE	The FASTIO parameter cannot process tapes.	None necessary.
118	W	FASTIO PARAMETER WAS SPECIFIED BUT COULD NOT BE PROCESSED	FASTIO files must have the same record type and length, and must reside on mass storage.	Recheck file specifications.
151	W	NOT SORT, MERGE OR OUTPUT; SORT ASSUMED	In the SMFILE call, file disposition must be indicated as SORT, MERGE, or OUTPUT.	None, or change SMFILE call to indicate MERGE or OUTPUT.
152	C	RETURNED FROM OWNCODE WITHOUT CALLING SMRTN	Control has returned to Sort/Merge from an owncode routine without using a call to SMRTN; processing terminates.	Determine the proper method of calling SMRTN.
153	C	EXCESSIVE NUMBER OF ERRORS FOUND (n). PROCESSING TERMINATED.	The maximum number of errors for one call to Sort/Merge have occurred; processing terminates.	Correct errors and resubmit job.
154	F	SMRTN CANNOT BE USED WHEN USING SM5 PROCEDURES	The Sort/Merge 4 compatible routine SMRTN has been called by a set of owncode procedures using SM5xxx calls.	Change SMRTN to RETURN or the equivalent.
155	C	RETURN CODE MUST BE IN RANGE 0-3	The return code from all owncode routines must be 0, 1, 2, or 3.	Change the value of the return code.

GLOSSARY C

This glossary defines terms unique to the description of Sort/Merge; common terms that have special connotations within the context of this manual are also included.

Ascending -
: The order of sorting keys so that the record having the numeric key with the highest value is written last on the output file, and character keys are sorted according to the specified collating sequence. See Sort Order.

ASCII -
: American National Standard Code for Information Interchange. An 8-bit code representing a prescribed set of 128 characters. Control Data operating systems use a 6-bit display code to represent a subset (called ASCII6) of these characters.

Basic Access Methods (BAM) -
: A file manager that processes sequential and word addressable file organizations. See CYBER Record Manager.

Beginning-of-Information (BOI) -
: The start of your first record in a file. System information, such as tape labels of sequential files, can appear before beginning-of-information.

Block -
: A logical or physical grouping of records. CYBER Record Manager defines block types I, C, K, and E for sequential files.

Byte -
: A group of bits. In the body of this manual, a byte is six bits that represents a single character.

Character -
: A letter, digit, punctuation mark, or mathematical symbol forming part of one or more of the standard character sets.

Collating Sequence -
: A sequence in which the characters that are acceptable to a computer are ordered for purposes of sorting, merging, and comparing.

Control Statement Section -
: The first section of a job; contains the sequence of control statements that specifies all steps for job execution.

Control Statement Sort -
: A sort performed on the basis of parameters in the SORT5 control statement and in optional directive files.

CYBER Control Language (CCL) -
: A language that allows you to insert a set of statements into the control statement section of your job; provides a means for you to determine the status of files, initiate tests and transfers within the control statement section, and display results in your job dayfile.

CYBER Record Manager (CRM) -
: A generic term relating to the common products Advanced Access Methods and Basic Access Methods that run under the NOS and NOS/BE operating systems and that allow a variety of record types, blocking types, and file organizations to be created and processed. The execution time input/output of COBOL, FORTRAN, Sort/ Merge, ALGOL, and the DMS-170 products is implemented through CYBER Record Manager. The input/output of the NOS and NOS/BE operating system utilities such as COPY or SKIPF are not implemented through CYBER Record Manager. All CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines.

Default -
: A value supplied by Sort/Merge when you omit a specification from a parameter list.

Descending -
: The order of sorting keys so that the record having the numeric key with the lowest value is written last on the output file, and character keys are sorted in reverse order of the specified collating sequence. See Sort Order.

Dialog -
: An interactive aid designed to guide an inexperienced user through a sort or merge specification.

Directive File -
: A file that can be used in a control statement sort or merge; contains processing parameters in addition to the parameters specified in the SORT5 or MERGE control statement. This file must be a sequential file of block type C and record type Z.

Display Code -
: A 6-bit code representing a 63-character or 64-character computer character set.

EBCDIC -
: The abbreviation for extended binary-coded decimal interchange code. Control Data operating systems use a 6-bit version (called EBCDIC6) to represent this code.

End-of-Information (EOI) -
: CYBER Record Manager defines end-of-information in sequential files in terms of the file residence, as shown in table C-1.

Entry Point -
: A location within a program that can be referenced from other programs. Each entry point has a unique name with which it is associated.

TABLE C-1. END-OF-INFORMATION BOUNDARIES

File Organization	File Residence	Physical Position
Sequential	Mass storage	After the last user record.
Sequential	Labeled tape in SI, I, S, or L format	After the last user record and before any file trailer labels.
Sequential	Unlabeled tape in SI or I format	After the last user record and before any file trailer labels.
Sequential	Unlabeled tape in S or L format	Undefined.

File -
A collection of records treated as a unit; represented by a logical file name and delimited by beginning-of-information and end-of-information.

FILE Control Statement -
A control statement that supplies file information table (FIT) values after a source language program is compiled or assembled but before the program is executed. Basic file characteristics such as organization, record type, and description can be specified in the FILE control statement.

File Information Table (FIT) -
A table through which your program communicates with BAM. All file processing executes on the basis of information in the FIT. You can set FIT fields directly or use parameters in a file access call that sets the fields indirectly.

Floating-Point Number -
Numeric data stored internally as binary values (corresponds to a FORTRAN REAL or COBOL COMPUTATIONAL-2 number).

Graphic -
A character that can be printed or displayed.

Integer -
Numeric data stored internally as a binary value rather than a character value; in a parameter or procedure call format, a number that does not include any character positions to the right of the assumed decimal point and is not signed.

Interactive -
Job processing in which you and the system communciate with each other, rather than your submitting a job and receiving output, having no control over the job while processing occurs.

Interactive Facility (IAF) -
The network host software product that allows you to enter commands and to communicate with an executing program from a time-sharing terminal; operates under NOS.

INTERCOM -
The software that directs the flow of information and data between the central site computer and a number of remote terminals; operates under NOS/BE.

Job -
A set of control statements and the data and directives used by those control statements. A batch job begins with the job, USER, and CHARGE statements on NOS and the job and ACCOUNT statements on NOS/BE. An interactive job begins with login to a terminal.

Key Type -
The name of a numeric data format or collating sequence.

Keyword -
A part of a parameter that you enter exactly as shown in the parameter format. It is not changed by any information you supply.

Library -
A collection of programs or routines that is searched by the loader for entry points referenced by a program.

Literal -
A constant completely defined by its own identity.

Local File -
A file currently assigned to a job, or a temporary file other than the primary file; often contains a copy of an indirect access file or data from a magnetic tape.

Logical File Name (lfn) -
The one to seven coded letters or digits by which the operating system recognizes a file. Every logical file name in a job must be unique and begin with a letter.

Logical Record -
Under NOS, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. Equivalent to a system-logical-record under NOS/BE.

Login -
The process used at a terminal to gain access to the system.

Logout -
The process used to end a terminal session.

Major Sort Key -
A sort key that is considered to be the most important and is specified first. See Sort Key, Minor Sort Key.

Mass Storage -
A disk pack or other rotating mass storage device; not a magnetic tape.

Merge -
The process of combining two or more presorted files.

Minor Sort Key -
A sort key that is considered of lesser importance and is specified after the most important sort key. See Sort Key, Major Sort Key.

Owncode Routine -
A subroutine you write that provides the capability of inserting, substituting, modifying, or deleting input and output records during Sort/Merge processing.

Parameter -
A variable identified by a keyword and assigned specific values in the SORT5 or MERGE control statement or in a directive file; parameters and their specified values direct Sort/Merge processing.

Partition -
CYBER Record Manager defines a partition as a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence, as shown in table C-2.

Notice that in a file with W type records, a short physical record unit (PRU) of level 0 terminates both a section and a partition.

Physical Record Unit (PRU) -
Under NOS and NOS/BE, the amount of information transmitted by a single physical operation of a specified device. The size of a PRU depends on the device, as shown in table C-3.

A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

Procedure -
A subroutine that returns values through its parameters; invoked when the name of the procedure is referenced in a CALL or ENTER statement.

Record -
CYBER Record Manager defines a record as a group of related characters. A record or a portion thereof is the smallest collection of information that can be passed between CRM and your program. Eight different record types exist, as defined by the RT field of the file information table.

Record Type -
The term record type can have one of several meanings, depending on the context. CYBER Record Manager defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

Relocatable -
An object program that can reside in any part of central memory. The actual starting address is established at load time.

TABLE C-2. PARTITION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Boundary
PRU device	W	I	A short PRU of level 0 containing a one-word deleted record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A short PRU of level 0 containing a control word with a flag indicating a partition boundary.
	D,F,R, T,U,Z	C	A short PRU of level 0 followed by a zero-length PRU of level 17_8.
	S	-	A zero-length PRU of level number 17_8.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary.
	D,F,T, R,U,Z	C,K,E	A tapemark.
	S	-	A tapemark.
Any other tape format	-	-	Undefined.

TABLE C-3. PRU SIZES

Device	Size in Number of 60-Bit Words
Mass storage.	64
Tape in SI format with binary data (NOS/BE only).	512
Tape in I format (NOS only).	512
Tape in any other format.	Undefined.

Result Array -
A means of passing statistics and results of a sort or merge back to a calling procedure.

Rewind -
An operation that positions a file at the beginning-of-information.

Section -
CYBER Record Manager defines a section as a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary, as shown in table C-4.

The NOS and NOS/BE operating systems equate a section with a system-logical-record of level 0 through 16_8.

Sequential File -
A file with records in the physical order in which they were written. No logical order exists other than the relative physical record position.

Signed Numeric Data -
Integer data stored internally in display code; sorted according to numeric order and sign of the integer the display code represents.

Sort Key -
A field of information within each record in a sort or merge input file that is used to determine the order in which records are written to the output file.

Sort Order -
The order for sorting keys, either ascending or descending.

System-Logical-Record -
Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure. Equivalent to a logical record under NOS.

Word -
A group of bits (or 6-bit characters) between boundaries imposed by the computer system. A word is 60 bits in length (10 characters).

TABLE C-4. SECTION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Representation
PRU device	W	I	A deleted one-word record pointing back to the last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0.
	W	C	A control word with flags indicating a section boundary. The control word is in a short PRU of level 0.
	D,F,R, T,U,Z	C	A short PRU with a level less than 17_8.
	S	-	Undefined.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with flags indicating a section boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with flags indicating a section boundary.
	D,F,R, T,U,Z	C,K,E	Undefined.
	S	-	Undefined.
Any other tape format	-	-	Undefined.

SORT/MERGE PARAMETER SUMMARY D

The format specifications for all parameters used in a SORT5 or a MERGE control statement are summarized and listed in this appendix. Detailed information for each parameter is referenced by page number and appears in section 3 of this manual.

The following abbreviations are used in this appendix:

char	character
expr	expression
lfn	logical file name
proc	procedure name
rep	repetition

	Page
DIALOG Parameter	3-4

```
DIALOG=YES (or DIA=Y)

DIALOG=NO  (or DIA=N)
```

DIR Parameter	3-3

DIR=lfn

DIR=(lfn_1,lfn_2,...)

E Parameter	3-4

```
E=lfn

E=$NULL
```

EL Parameter	3-4

```
EL=T

EL=W

EL=F

EL=C
```

ENR Parameter	3-4

```
ENR=expr

ENR=expr .. expr
```

FASTIO Parameter	3-9

```
FASTIO=YES (or FASTIO=Y)

FASTIO=NO  (or FASTIO=N)
```

	Page
FROM Parameter	3-2

FROM=lfn

FROM=(lfn_1,lfn_2,...)

FROM=$NULL

KEY Parameter	3-2

$$\text{KEY=}\begin{bmatrix}\text{(key\_def[,key\_def]...)}\\ \text{range}\end{bmatrix}$$

where:

$$\text{key\_def=}\begin{bmatrix}\text{range}\\ \text{(range[,type [,ad]])}\\ \text{(first,length [,type [,ad]])}\end{bmatrix}$$

$$\text{range = }\begin{bmatrix}\text{first}\\ \text{first..last}\end{bmatrix}$$

L Parameter	3-3

```
L=lfn

L=$NULL
```

OWNF Parameter	3-4

```
OWNF=lfn
```

OWNFL Parameter	3-5

```
OWNFL=integer

OFL=integer
```

OWNMRL Parameter	3-5

```
OWNMRL=integer

OMRL=integer
```

OWNn Parameter	3-5

```
OWNn=proc
```

RETAIN Parameter	3-5

```
RETAIN=YES (or RET=Y)

RETAIN=NO  (or RET=N)
```

	Page
SEQx Parameters	3-5

```
    SEQN=name
    SEQS=('char',...,'char')
    .
    .
    .
    [SEQR=YES OR SEQR=Y]
    [SEQA=YES OR SEQA=Y]
```

STATUS Parameter	3-7

```
    STATUS=variable (or ST=variable)
```

SUM Parameter	3-8

```
    SUM=((sum_def)[,(sum_def)]...)

  where:

    sum_def= [(first,length,type[,rep])]
             [(first..last,type[,rep]) ]
```

	Page
TO Parameter	3-2

```
    TO=lfn

    TO=$NULL
```

VERIFY Parameter	3-9

```
    VERIFY=YES (or VER=Y)

    VERIFY=NO (or VER=N)
```

INTERACTIVE DIALOG E

This appendix contains the text of the interactive dialog (figure E-1) that is invoked when you specify DIALOG=YES. The questions asked by Sort/Merge are indicated by uppercase letters. The answers you supply are indicated by lowercase letters; "next" or a number following an answer indicates the next question that is asked by Sort/Merge. For questions requiring yes or no answers, you can abbreviate and use y or n.

```
ARE YOU SURE THAT ALL YOUR FILES (IF ANY) ARE EITHER STANDARD UNIT RECORD FORMAT OR HAVE HAD FILE
CONTROL STATEMENTS SPECIFIED FOR THEM?
yes (1)
no  (next)

UNLESS YOU SPECIFY OTHERWISE, ALL INPUT FILES AND THE OUTPUT FILE ARE ASSUMED TO HAVE STANDARD UNIT
RECORD FORMAT SUCH AS LINES TYPED FROM A TERMINAL, LINES IN A BATCH JOB, OR LINES TO BE PRINTED ON A
PRINTER OR TERMINAL.  THESE ARE TECHNICALLY KNOWN AS BT=C, RT=Z, FL=N WITH N NO MORE THAN 150.  DO YOU
HAVE ANY FILES THAT ARE NOT OF THIS FORMAT?
yes (next)
no  (1)

FOR EACH FILE THAT IS NOT OF THIS FORMAT YOU MUST HAVE SPECIFIED A FILE CONTROL STATEMENT.  SORT/MERGE
WILL QUIT NOW SO YOU MAY DO SO.

1.  DO YOU HAVE JUST ONE INPUT FILE AND NO OWN1 OR OWN2 SUBROUTINES?
    yes       (next)
    no        (6)

2.  WHAT IS THE NAME OF YOUR INPUT FILE?
    file-name

3.  DO YOU HAVE ANY OWN3 OR OWN4 SUBROUTINES?
    yes       (9)
    no        (next)

4.  WHAT IS THE NAME OF YOUR OUTPUT FILE?
    file-name

5.  THE STANDARD COLLATING SEQUENCE IS ASCII6.  HERE IS THE WHOLE ASCII6 COLLATING SEQUENCE LISTED
     !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
    DO ALL OF YOUR KEYS CONSIST OF CHARACTER DATA TO BE SORTED ACCORDING TO THE ASCII6 COLLATING
    SEQUENCE?
    yes       (next)
    no        (12)

    ASCENDING ORDER OF THE ASCII6 COLLATING SEQUENCE MEANS THE NORMAL ORDER.
     !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
    DESCENDING ORDER MEANS THE REVERSE ORDER.
    _^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#"! 
    DO YOU WANT ALL YOUR KEYS SORTED IN ASCENDING ORDER?
    yes       (next)
    no        (next)

33. HOW MANY KEY FIELDS DO YOU HAVE?
    number

    WHAT IS THE TYPE OF THE MOST IMPORTANT KEY FIELD?
    key-type

    THE LEFT-MOST CHARACTER OF A RECORD IS NUMBER 1.  FOR THE MOST IMPORTANT KEY FIELD, WHAT IS THE
    NUMBER OF THE LEFT-MOST CHARACTER?
    number

    HOW MANY CHARACTERS ARE IN THE MOST IMPORTANT KEY FIELD?
    number
```

Figure E-1. Interactive Dialog (Sheet 1 of 5)

```
    DO YOU WANT THIS KEY FIELD SORTED IN ASCENDING ORDER?†
    yes      (next)
    no       (next)

    FOR THE SECOND†† MOST IMPORTANT KEY FIELD, WHAT IS THE TYPE OF THE KEY?
    key-type

    FOR THE SECOND†† MOST IMPORTANT KEY FIELD, WHAT IS THE NUMBER OF THE LEFT-MOST CHARACTER?
    number

    HOW MANY CHARACTERS ARE IN THE SECOND†† MOST IMPORTANT KEY FIELD?
    number

    DO YOU WANT RECORDS WITH EQUIVALENT KEY VALUES TO BE WRITTEN IN THE SAME ORDER THEY ARE READ?
    yes      (next)
    no       (next)

    A SUM FIELD IS A SPECIAL FIELD IN A RECORD THAT IS CHANGED BY SORT/MERGE.  IF SUM FIELDS ARE
    SPECIFIED SUM FIELDS FROM RECORDS WITH EQUAL KEY VALUES WILL BE NUMERICALLY SUMMED AND PUT IN THE
    SAME SUM FIELD IN A NEW RECORD.  THE REMAINING KEY AND DATA FIELDS IN THE NEW RECORD WILL BE SET TO
    THE CORRESPONDING FIELDS IN ONE OF THE OLD RECORDS.  THE ORIGINAL RECORDS WILL BE DELETED.  RECORDS
    MAY HAVE MORE THAN ONE SUM FIELD, AND THEY WILL BE SORTED ACCORDING TO KEY VALUES.  THEY TYPE OF A
    SUM MAY BE ANY NUMERIC KEY TYPE EXCEPT REAL.

28. HOW MANY SUM FIELDS DO YOU HAVE?
    number

    WHAT IS THE TYPE OF THE FIRST†† SUM FIELD?
    sum-type

    WHAT IS THE FIRST CHARACTER IN THE FIRST†† SUM FIELD?
    number

    HOW MANY CHARACTERS‡ LONG IS THE FIRST†† SUM FIELD?
    number

    GIVE AN ESTIMATE OF THE MAXIMUM NUMBER OF RECORDS TO BE SORTED.
    number   (end of dialog)

6.  DO YOU HAVE AN OWN1 SUBROUTINE?
    yes      (next)
    no       (7)

    WHAT IS THE NAME OF THE FILE CONTAINING YOUR OWNCODE SUBROUTINE(S)?
    file-name

    WHAT IS THE NAME OF YOUR OWN1 SUBROUTINE?
    entry-name

7.  DO YOU HAVE AN OWN2 SUBROUTINE?
    yes      (next)
    no       (8)

    WHAT IS THE NAME OF THE FILE CONTAINING YOUR OWNCODE SUBROUTINE(S)?‡‡
    file-name

8.  HOW MANY INPUT FILES DO YOU HAVE?
    1        (2)
    number   (next)

    WHAT IS THE NAME OF YOUR FIRST†† INPUT FILE?
    file-name

9.  WHAT IS THE NAME OF THE FILE CONTAINING YOUR OWNCODE SUBROUTINE(S)?‡‡
    file-name

    DO YOU HAVE AN OWN3 SUBROUTINE?
    yes      (next)
    no       (10)
```

Figure E-1. Interactive Dialog (Sheet 2 of 5)

```
    WHAT IS THE NAME OF YOUR OWN3 SUBROUTINE?
    entry-name

    DO YOU HAVE AN OWN4 SUBROUTINE?
    yes       (next)
    no        (11)

10. WHAT IS THE NAME OF YOUR OWN4 SUBROUTINE?
    entry-name

11. DO YOU HAVE AN OUTPUT FILE?
    yes       (4)
    no        (5)

12. DO YOU KNOW THE NAMES OF THE KEY TYPES YOU WANT TO USE?
    yes       (33)
    no        (next)

    DO YOUR RECORDS CONSIST ENTIRELY OF CHARACTERS?
    yes       (19)
    no        (next)

    NON-CHARACTER DATA MAY BE ORDERED ACCORDING TO ITS BINARY (SIGNED OR UNSIGNED) VALUE, OR REAL (I.E.
    FLOATING POINT) VALUE.

    A SIGNED BINARY KEY INTERPRETS THE BITS WITHIN THE KEY FIELD AS A ONE'S COMPLEMENT BINARY NUMBER.
    IF THE LEFT-MOST BIT IS ZERO THEN THE NUMBER IS POSITIVE.  IF THE LEFT-MOST BIT IS ONE THEN THE
    NUMBER IS NEGATIVE AND THE MAGNITUDE OF THE NUMBER IS THAT WITH ALL BITS COMPLEMENTED.  IF THE KEY
    FIELD STARTS AND ENDS ON A CHARACTER BOUNDARY THEN THE KEY TYPE IS "INTEGER", OTHERWISE IT IS
    "INTEGER_BITS".

    AN UNSIGNED BINARY KEY INTERPRETS THE BITS WITHIN THE KEY FIELD AS A BINARY NUMBER.  THE LEFT-MOST
    BIT IN THE FIELD CONTRIBUTES TO THE NUMERIC VALUE JUST LIKE ANY OTHER BIT IN THE FIELD.  IF THE KEY
    FIELD STARTS AND ENDS ON A CHARACTER BOUNDARY THEN THE KEY TYPE IS "BINARY", OTHERWISE IT IS
    "BINARY_BITS".

    A REAL OR FLOATING POINT KEY INTERPRETS A 60-BIT VALUE AS A STANDARD FLOATING POINT NUMBER.  THE KEY
    TYPE IS "REAL".

13. IF YOU WANT MORE INFORMATION ABOUT A KEY TYPE, INCLUDING HOW TO SPECIFY IT, TYPE THE NAME OF THE KEY
    TYPE.  (OTHERWISE, TYPE A SPACE.)
    binary     (14)
    binary_bits (15)
    integer    (16)
    integer_bits (17)
    real       (18)
    Δ          (19)

14. A BINARY KEY FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE LENGTH OF THE FIELD.
    (THE FIRST CHARACTER POSITION IN THE RECORD IS 1.)
    (13)

15. A BINARY_BITS KEY FIELD IS SPECIFIED BY NAMING THE FIRST BIT POSITION AND THE LENGTH OF THE FIELD.
    (THE FIRST BIT POSITION IN THE RECORD IS 1.)
    (13)

16. AN INTEGER KEY FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE LENGTH OF THE
    FIELD.  (THE FIRST CHARACTER POSITION IN THE RECORD IS 1.)
    (13)

17. AN INTEGER_BITS KEY FIELD IS SPECIFIED BY NAMING THE FIRST BIT POSITION AND THE BIT LENGTH OF THE
    FIELD.  (THE FIRST BIT POSITION IN THE RECORD IS 1.)
    (13)

18. A REAL KEY FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE CHARACTER LENGTH OF THE
    FIELD.  SINCE REAL KEY FIELDS MUST BE FULL WORDS AND ALIGNED ON WORD BOUNDARIES, THE FIRST CHARACTER
    POSITION MUST SPECIFY THE FIRST CHARACTER OF A WORD AND THE LENGTH MUST BE 10.
    (13)
```

Figure E-1. Interactive Dialog (Sheet 3 of 5)

```
19. KEY FIELDS CONSISTING ONLY OF CHARACTERS MAY BE ORDERED ACCORDING TO A CHARACTER COLLATING SEQUENCE
    OR ACCORDING TO A NUMERIC INTERPRETATION OF THE CHARACTERS.

    DO YOU WANT ANY OF YOUR CHARACTER KEY FIELDS TO BE INTERPRETED NUMERICALLY?
    yes      (next)
    no       (27)

    THE KEY TYPE FOR NUMERIC CHARACTER KEYS IS "NUMERIC_XX" WHERE XX INDICATES HOW THE SIGN (IF ANY) IS
    TO BE EXTRACTED AND WHETHER LEADING BLANKS ARE ALLOWED.  THE LEGAL NUMERIC CHARACTER KEY TYPES ARE:
    NUMERIC_NS      ALL DIGITS, NO SIGN
    NUMERIC_FS      LEADING BLANKS, THEN OPTIONAL NEGATIVE SIGN, THEN DIGITS
    NUMERIC_LS      REQUIRED SIGN IS FIRST, THEN DIGITS
    NUMERIC_TS      DIGITS FOLLOWED BY REQUIRED SIGN
    NUMERIC_LO      LEADING OVERPUNCHED DIGIT, THEN DIGITS
    NUMERIC_TO      DIGITS FOLLOWED BY TRAILING OVERPUNCHED DIGIT

20. IF YOU WANT MORE INFORMATION ABOUT ONE OF THE NUMERIC CHARACTER KEY TYPES, INCLUDING HOW TO SPECIFY
    IT, TYPE THE NAME OF THE KEY TYPE.  (OTHERWISE, TYPE A SPACE.)
    numeric_ns (21)
    numeric_fs (22)
    numeric_ls (23)
    numeric_ts (24)
    numeric_lo (25)
    numeric_to (26)
    Δ          (27)

21. NUMERIC_NS KEY FIELDS MUST CONTAIN ALL DECIMAL DIGIT CHARACTERS WITHOUT A SIGN.  THE FIELD IS
    SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE CHARACTER LENGTH OF THE FIELD.
    (20)

22. NUMERIC_FS KEY FIELDS MUST BE NUMERIC CHARACTERS IN A FLOATING SIGN FORMAT.  LEADING BLANKS ARE
    REQUIRED FOR LEADING ZEROS, FOLLOWED BY AN OPTIONAL SIGN CONSISTING OF "-", FOLLOWED BY DECIMAL
    DIGIT CHARACTERS.  THIS IS THE SAME FORMAT AS IS PRODUCED BY WRITING ACCORDING TO THE DEFAULT
    FORTRAN I FORMAT WITH NO MODIFIERS.  THE FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION
    AND THE CHARACTER LENGTH OF THE FIELD.
    (20)

23. THE FIRST CHARACTER OF A NUMERIC_LS KEY FIELD MUST BE A "+" OR "-" CHARACTER.  THE REMAINING
    POSITIONS MUST CONSIST OF DECIMAL DIGIT CHARACTERS.  THE FIELD IS SPECIFIED BY NAMING THE FIRST
    CHARACTER POSITION AND THE CHARACTER LENGTH OF THE FIELD.
    (20)

24. THE TRAILING (RIGHT-MOST) CHARACTER OF A NUMERIC_TS KEY FIELD MUST BE A "+" OR "-" CHARACTER.  THE
    REMAINING POSITIONS MUST CONSIST OF DECIMAL DIGIT CHARACTERS.  THE FIELD IS SPECIFIED BY NAMING THE
    FIRST CHARACTER POSITION AND THE CHARACTER LENGTH OF THE FIELD.
    (20)

25. THE LEADING (LEFT-MOST) CHARACTER OF A NUMERIC_LO KEY FIELD MUST BE AN OVERPUNCHED DIGIT.  THE
    OVERPUNCHED DIGIT DETERMINES BOTH THE SIGN OF THE ENTIRE NUMBER AND THE VALUE OF THE LEADING DIGIT.
    CHARACTERS FROM "0" TO "9" MEAN THE SIGN IS POSITIVE AND THE LEADING DIGIT IS 0 TO 9.  CHARACTERS
    FROM "A" TO "I" MEAN THE SIGN IS POSITIVE AND THE LEADING DIGIT IS 1 to 9.  CHARACTERS FROM "J" TO
    "R" MEAN THE SIGN IS NEGATIVE AND THE LEADING DIGIT IS 1 TO 9.  THE CHARACTER "<" MEANS THE SIGN IS
    POSITIVE AND THE LEADING DIGIT IS 0.  THE CHARACTER "!" MEANS THE SIGN IS NEGATIVE AND THE LEADING
    DIGIT IS ZERO.  THE REMAINING POSITIONS OF THE FIELD MUST CONTAIN DECIMAL DIGITS.  THE FIELD IS
    SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE CHARACTER LENGTH OF THE FIELD.
    (20)

26. THE TRAILING (RIGHT-MOST) CHARACTER OF A NUMERIC_TO KEY FIELD MUST BE AN OVERPUNCHED DIGIT.  THE
    OVERPUNCHED DIGIT DETERMINES BOTH THE SIGN OF THE ENTIRE NUMBER AND THE VALUE OF THE TRAILING
    DIGIT.  CHARACTERS FROM "0" TO "9" MEAN THE SIGN IS POSITIVE AND THE TRAILING DIGIT IS 0 TO 9.
    CHARACTERS FROM "A" TO "I" MEAN THE SIGN IS POSITIVE AND THE TRAILING DIGIT IS 1 TO 9.  CHARACTERS
    FROM "J" TO "R" MEAN THE SIGN IS NEGATIVE AND THE TRAILING DIGIT IS 1 TO 9.  THE CHARACTER "<" MEANS
    THE SIGN IS POSITIVE AND THE LEADING DIGIT IS 0.  THE CHARACTER "!" MEANS THE SIGN IS NEGATIVE AND
    THE TRAILING DIGIT IS ZERO.  THE REMAINING POSITIONS OF THE FIELD MUST CONTAIN DECIMAL DIGITS.  THE
    FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE CHARACTER LENGTH OF THE FIELD.
    (20)
```

Figure E-1. Interactive Dialog (Sheet 4 of 5)

```
27. DO YOU WANT ANY OF YOUR CHARACTER KEYS TO BE ORDERED ACCORDING TO A COLLATING SEQUENCE?
    yes       (next)
    no        (32)

    THE STANDARD COLLATING SEQUENCE IS ASCII6.  HERE IS THE WHOLE ASCII6 COLLATING SEQUENCE LISTED
     !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_

    THE KEY TYPE OF A CHARACTER KEY FIELD ORDERED ACCORDING TO A COLLATING SEQUENCE IS THE NAME OF THE
    COLLATING SEQUENCE.  THE KEY FIELD IS SPECIFIED BY NAMING THE FIRST CHARACTER POSITION AND THE
    LENGTH OF THE FIELD.

    THERE ARE FIVE PRE-DEFINED COLLATING SEQUENCES:  DISPLAY, COBOL6, ASCII6, EBCDIC6 AND INTBCD.  THERE
    CAN BE FROM 0 TO 100 USER-DEFINED COLLATING SEQUENCES.
    DO YOU WANT TO DEFINE YOUR OWN COLLATING SEQUENCE?
    yes       (next)
    no        (32)

29. WHEN ASKED TO TYPE IN YOUR COLLATING SEQUENCE PLEASE TYPE IT IN AS C,C,C,C,C,C OR C,C,C,C,
                                                                                         C,C
    WHERE C IS A CHARACTER OR OCTAL VALUE.
    C CAN BE A LETTER, A ONE-DIGIT NUMBER OR A SPECIAL CHARACTER SUCH AS ( ) + - *.
    SORT/MERGE ASSUMES A TWO-DIGIT NUMBER IS AN OCTAL VALUE.  (THE NUMBER SHOULD NOT BE FOLLOWED BY A
    LETTER "B".)  THE COMMA HAS TO BE SPECIFIED BY ITS OCTAL VALUE 56.  HOW MANY COLLATING SEQUENCES DO
    YOU WANT TO DEFINE?
    number

30. WHAT IS THE NAME OF YOUR COLLATING SEQUENCE?
    sequence-name

    PLEASE TYPE IN YOUR COLLATING SEQUENCE.
    c,c,...
    (30 if more to define)
    (next if done defining)

    DEFINING EQUATE MEANS SPECIFYING TWO OR MORE CHARACTERS ALREADY IN A COLLATING SEQUENCE AS EQUAL FOR
    COMPARISON PURPOSES.  MORE THAN ONE EQUATE COLLATING SEQUENCE MAY BE SPECIFIED FOR A COLLATING
    SEQUENCE.
    DO YOU WANT TO DEFINE EQUATES FOR YOUR COLLATING SEQUENCES?
    yes       (next)
    no        (32)

    WHEN ASKED TO TYPE IN YOUR EQUATE SEQUENCE(S), PLEASE ENTER IN THE FORM C,C,C,C,C,C WHERE C IS A
    CHARACTER OR OCTAL VALUE AS EXPLAINED BEFORE.
    HOW MANY EQUATE SEQUENCES DO YOU WANT TO DEFINE?
    number

31. WHAT IS THE NAME OF THE COLLATING SEQUENCE FOR THIS EQUATE?
    sequence-name

    PLEASE TYPE IN YOUR EQUATE SEQUENCE.
    c,c,...
    (31 if more to define)
    (next if done defining)

32. YOU SHOULD KNOW THE NAME(S) OF ALL KEY TYPES YOU WANT TO USE.  IF YOU DO NOT KNOW, START OVER BY
    TYPING "RESTART" INSTEAD OF THE NUMBER OF KEY FIELDS.
    (33)
```

†Not asked if you answered "yes" to the previous question about sort order.

††Repeated with successive numbers until all have been specified.

‡"BITS" if you specified sum type INTEGER_BITS or BINARY_BITS.

‡‡Not asked if you gave the file name in response to question 6.

Figure E-1. Interactive Dialog (Sheet 5 of 5)

SORT/MERGE 5, SORT/MERGE 4 COMPARISON F

Sort/Merge 5 is functionally equivalent to Sort/Merge 4, but these versions of the product are externally incompatible because the control statement for Sort/Merge 5 has been changed. Table F-1 shows a comparison of Sort/Merge 5 and Sort/Merge 4. As indicated in table F-1, the Sort/Merge 4 FORTRAN procedures can be called by Sort/ Merge 5 users, and the Sort/Merge 4 owncode exits can be taken from FORTRAN 5.

TABLE F-1. SORT/MERGE 5, SORT/MERGE 4 COMPARISON

Feature	Sort/Merge 5	Sort/Merge 4
Control Statement Sort	Most sorts can be specified with the SORT5 control statement and parameters alone.	A control statement sort requires a separate directives section of the input file.
Merge	Merge-only processing is available.	Merge-only processing is available.
Key Types	Sort keys can be 6-bit display code characters, signed or unsigned binary integers, or floating-point numbers.	Sort keys can be 6-bit characters written in display code or internal BCD code, signed or unsigned binary integers, or floating-point numbers.
Collating Sequences	ASCII6 (default), COBOL6, DISPLAY, EBCDIC6, and INTBCD are predefined. User-defined is also available.	ASCII6 (default if installation character set is ASCII), COBOL6 (default if installation character set is CDC), DISPLAY, and INTBCD are predefined. User-defined is also available.
Record Types	Any CYBER Record Manager sequential record type except U can be sorted.	Any CYBER Record Manager sequential record type except U can be sorted.
Tape Sorts	Available in a future release.	A tape variant provides balanced and polyphase tape sorting.
Checkpoint/ Restart	Not available.	A job abnormally terminated can be captured on tape so that the job can be restarted from the checkpoint, rather than from the beginning.
Interactive Usage	Interactive dialog and commands are available to aid in sort or merge specification.	No interactive dialog or commands are available.
Procedure Calls	Sort or merge can be initiated from within a program (FORTRAN or other language) by a series of procedure calls.	Sort or merge can be initiated from within a COMPASS or FORTRAN program by a series of procedure calls. The Sort/Merge 4 FORTRAN procedures can be called by Sort/Merge 5 users.
Owncode Routines	Routines can be written to process input records, output records, or records with equal keys during Sort/Merge processing.	Routines can be written to process input records, output records, records with equal keys, or nonstandard labels during Sort/ Merge processing. The Sort/Merge 4 FORTRAN owncode exits can be taken by Sort/Merge 5 users.
COBOL	COBOL5 control statement option can specify Sort/Merge 5 be used when COBOL SORT or MERGE statement is executed, or direct calls to the procedures can be made.	The interface is through the COBOL SORT or MERGE statement; Sort/Merge 4 is the default for sorts or merges from COBOL 5.

INDEX

COMMENT SHEET

MANUAL TITLE: Sort/Merge Version 5 Reference Manual

PUBLICATION NO.: 60484800

REVISION: C

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments on the back (please include page number references).

_____ Please reply _____ No reply necessary

FOLD FOLD

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
P.O. BOX 3492
Sunnyvale, California 94088-3492

CUT ALONG LINE

FOLD FOLD

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.
FOLD ON DOTTED LINES AND TAPE

NAME:

COMPANY:

STREET ADDRESS:

CITY/STATE/ZIP:

TAPE TAPE