SERIES 60 (LEVEL 6)

GCOS 6 SORT/MERGE ADDENDUM B

SUBJECT

Changes and Additions to the Manual

SPECIAL INSTRUCTIONS

This update is the second addendum to CB04, Rev. 0, dated January 1978. Insert the attached pages into the manual according to the collating instructions on the back of this cover. Change bars in the margin indicate technical changes and additions; asterisks denote deletions.

Note:

Insert this cover behind the manual cover as evidence the manual is updated with Addendum B.

ORDER NUMBER

CB04B, Rev. 0

November 1978

22101
21178
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update this manual, remove old pages and insert new pages as follows:

Remove	Insert
Title Page, Preface	Title Page, Preface
iii through vi	iii through iv
1-1, 1-2	1-1, 1-2
2-3 through 2-6	2-3 through 2-6
3-1 through 3-3	3-1, 3-2
	3-3, blank
4-1, 4-2	4-1, 4-2
5-3, blank	5-3, blank
6-1 through 6-6	6-1 through 6-6
7-1, 7-2	7-1, 7-2
7-7 through 7-12	7-7 through 7-12
A-1, A-2	A-1 through A-5

File No.: 1S53

11/78
CB04B

SERIES 60 (LEVEL 6)

GCOS 6 SORT/MERGE

SUBJECT

Detailed Description of the Series 60 (Level 6) GCOS 6 Sort/Merge

SOFTWARE SUPPORTED

This manual describes Release 0305 of the GCOS 6 Sort/Merge which executes under the control of the following operating systems:

- GCOS 6 MOD 200 (Release 0100)
- GCOS 6 MOD 400 (Releases 0110 and 0120)
- GCOS 6 MOD 600 (Release 0100)

When a later release of any operating system occurs, see the Manual Directory in the appropriate *System Concepts* manual to determine if this version of the manual applies to that release.

ORDER NUMBER

CB04, Rev. 0

January 1978

Honeywell

Preface

This manual describes the GCOS 6 Sort/Merge. Unless stated otherwise, the term GCOS refers to the GCOS 6 software; the term Level 6 refers to the Series 60 (Level 6) hardware on which the software executes.

Section 1 summarizes the capabilities of the Sort and Merge programs.

Section 2 describes the Sort language, including the invoking SORT command and the Sort Description that particularizes the sort application.

Section 3 describes the Sort Report and error messages generated during execution of the Sort program.

Section 4 describes the Merge language.

Section 5 describes the Merge Report and error messages.

Section 6 describes the operating procedures for invoking Sort and Merge and submitting the Sort and Merge Descriptions. This section also includes sample sort and merge runs.

Section 7 explains sorting using subroutine calls.

Appendix A specifies Sort and Merge memory requirements.

Appendix B is the ASCII collating sequence.

Appendix C describes a debug mode.

File No.: 1S53 CB04

MANUAL DIRECTORY

The following publications constitute the GCOS 6 manual set. See the "Manual Directory" of the appropriate *System Concepts* manual for the current revision number and addenda (if any) of the relevant operating-system-specific publications.

Order No.	*Manual Title*
CB01	*GCOS 6 Program Preparation*
CB02	*GCOS 6 Commands*
CB03	*GCOS 6 Communications Processing*
CB04	*GCOS 6 Sort/Merge*
CB05	*GCOS 6 Data File Organizations and Formats*
CB06	*GCOS 6 System Messages*
CB07	*GCOS 6 Assembly Language Reference*
CB08	*GCOS 6 System Service Macro Calls*
CB09	*GCOS 6 RPG Reference*
CB10	*GCOS 6 Intermediate COBOL Reference*
CB11	*GCOS 6 BASIC Reference*
CB12	*GCOS 6 Entry-Level COBOL Reference*
CB13	*GCOS 6 Entry-Level FORTRAN Reference*
CB20	*GCOS 6 MOD 400 System Concepts*
CB21	*GCOS 6 MOD 400 Program Execution and Checkout*
CB22	*GCOS 6 MOD 400 Programmer's Guide*
CB23	*GCOS 6 MOD 400 System Building*
CB24	*GCOS 6 MOD 400 Operator's Guide*
CB27	*GCOS 6 MOD 400 Programmer's Pocket Guide*
CB28	*GCOS 6 MOD 400 Master Index*
CB30	*Remote Batch Facility User's Guide*
CB31	*Data Entry Facility User's Guide*
CB32	*Data Entry Facility Operator's Quick Reference Guide*
CB33	*Level 6/Level 6 File Transmission Facility User's Guide*
CB34	*Level 6/Level 62 File Transmission Facility User's Guide*
CB35	*Level 6/Level 64 (Native) File Transmission Facility User's Guide*
CB36	*Level 6/Level 66 File Transmission Facility User's Guide*
CB37	*Level 6/Series 200/2000 File Transmission Facility User's Guide*
CB38	*Level 6/BSC 2780/3780 File Transmission Facility User's Guide*
CB39	*Level 6/Level 64 (Emulator) File Transmission Facility User's Guide*
CB40	*2780/3780 Workstation Facility User's Guide*
CB41	*HASP Workstation Facility User's Guide*
CB42	*Level 66 Host Resident Facility User's Guide*
CB43	*Terminal Concentration Facility User's Guide*
CB44	*Interactive Function User's Guide*
CB48	*3270 Workstation Facility User's Guide*
CB50	*GCOS 6 MOD 600 System Concepts*
CB51	*GCOS 6 MOD 600 Program Execution and Checkout*

CB52	*GCOS 6 MOD 600 Programmer's Guide*
CB53	*GCOS 6 MOD 600 System Building*
CB54	*GCOS 6 MOD 600 Administrator's Guide*
CB55	*GCOS 6 MOD 600 Transaction Driven System*
CB56	*I-D-S/II Data Base Administrator's Guide*
CB57	*I-D-S/II Data Base User's Guide*
CB58	*GCOS 6 MOD 600 Operator's Guide*
CD46	*Display Formatting and Control*
CD47	*GCOS 6 MOD 200 System Concepts*
CD48	*GCOS 6 MOD 200 Application Development Guide*
CD49	*GCOS 6 MOD 200 Operator's Guide*
CD50	*GCOS 6 MOD 200 HASP Workstation Facility User's Guide*

In addition, the following publications provide supplementary information:

Order No.	*Manual Title*
AS22	*Honeywell Level 6 Minicomputer Handbook*
AT04	*Level 6 System and Peripherals Operation Manual*
AT97	*MLCP Programmer's Reference Manual*
FQ41	*Writable Control Store User's Guide*

Section 1
Sort and Merge Capabilities

The Sort and Merge utility programs provide file sorting and file merging capabilities, respectively. Sort arranges records from an input file in an order based on the values of record key fields defined by the user, and places the ranked records in the specified output file. Merge combines the records of up to six sequentially ordered input files. Information supplied by the user defines the specific function to be performed. The general characteristics of Sort and Merge are summarized in this section.

Note:

A sorting function for MOD 400 and MOD 600 may also be invoked from COBOL, FORTRAN and assembly language programs (see Section 7). A sorting function for MOD 200 may be invoked through TCL (transaction control language). See the *Application Development Guide* for details.

SUMMARY OF FEATURES APPLICABLE TO BOTH SORT AND MERGE

Sort and Merge incorporate these features:

- Up to 16 key fields may be specified; they may be contiguous, separated, or overlapped. Keys may be specified as being in ascending or descending order according to the ASCII collating sequence.
- Key field data types may be any of the following:
  - — Character string.
  - — Single- or double-word signed binary.
  - — Unpacked decimal: unsigned or signed with the sign leading, trailing, or trailing overpunched.
  - — Packed decimal: unsigned or signed trailing.
- Record selection (i.e., the inclusion or exclusion of records as input to Sort or Merge) is based on record content only or on record content and a specified value.
- Compound record selection permits LOGICAL AND and/or INCLUSIVE OR relationships.
- Output records may be rearrangements of input record fields.
- Commands and statements may be entered through a terminal, card reader, or disk file.
- Input and output files may have sequential, indexed, or relative organization.
- The input and output files need not have the same record attributes.
- Records with duplicate keys may be deleted.
- A report comprising statistical information is directed to the console or a printer.

SUMMARY OF FEATURES APPLICABLE ONLY TO SORT

Sort incorporates the following features:

- Sort uses a disk work file; the work file may be permanent or temporary.
- The work file uses a minimal amount of disk file space; i.e., it is approximately 1.2 times the size required to support the output file.
- The input, output, and work files may reside on the same device.
- Records with duplicate key fields can be ordered on a first-in, first-out (FIFO) basis.
- The output record may consist of the input record address followed by the key fields or it may consist only of the input record address (i.e., the ADDROUT file).

SUMMARY OF FEATURES APPLICABLE ONLY TO MERGE

Merge incorporates the following features:

- Up to six sequentially ordered input files may be merged.
- A single input file may be used, thus providing a file restructuring capability.
- Input files need not have the same file or record attributes, except for the key and record selection fields.
- If duplicate records are found (i.e., they have the same key field), the records are written to the output file in the order in which the files containing them appear in the input file list; this is the Merge FIFO rule.

SYSTEM REQUIREMENTS

The minimum system requirements are:

- 8K words of memory for execution of Sort or Merge.
- One KSR-like device.
- Disk work file for Sort.
- Devices for supported input and output files.

GENERAL DESCRIPTION OF SORT AND MERGE

Sort and Merge are utility programs that execute under either an online task group or a batch task group in the operating system environment.

The command SORT or MERGE is submitted to the command processor to invoke Sort or Merge, respectively.

Sort and Merge Descriptions are submitted to the Sort and Merge programs respectively; they designate which files will be used by Sort/Merge and the keys on which sorting/merging are to be based.

RECORD KEYS

Sort arranges records from an input file according to the values of record key fields, and places the ranked records in the specified output file. Merge combines the records of up to six sequentially ordered input files. Up to 16 key fields within each record can be used in ranking the input records: 1 major key field and 15 minor key fields. Records are ordered first according to the major key; then all records containing the same major key are sorted/merged according to the minor keys in the sequence dictated by the Sort/Merge Description.

RECORD SELECTION

Records can be selected or omitted as input to Sort or Merge through record selection. Record selection may be based on the meeting of conditions defined by comparison operations between two fields within a record or between a field of a record and a specified value. The conditions for record selection are specified in INCL or OMIT statements, which are described later in this manual. A maximum of four conditions may be specified; they may be within a single statement or be interspersed among up to four statements.

RECORD ARRANGEMENT

The bytes of the input record(s) that will constitute the output records, and the order in which these bytes will occur can be specified for Sort and Merge. Up to 16 byte string descriptions may be specified.

KEY SORT OUTPUT

Sort output can consist of a field giving the input record address and be followed by the sort keys or it can consist only of the input record address.

Sort Description cannot be divided between two lines or records (i.e., it cannot be started on one line or record and completed on the next line or record). Words of the Sort Description to be processed must be contained in the first 80 characters of a line.

Parentheses can be used to enclose any word or words in the statement to enhance readability.

Note:

In the statement format descriptions given below, where words are separated by a space, the space may be replaced with a comma.

NOTATIONAL SYMBOLS IN SORT DESCRIPTION

The following notational conventions are used in the format descriptions of statements:

Convention	Meaning
UPPERCASE CHARACTERS	Required word; must be used in the form specified.
lowercase characters	Symbolic name; must be replaced by user-supplied word or words.
Brackets [ ]	The item enclosed in the brackets is optional.
Braces { }	An enclosed entry must be selected.
Ellipses . . .	The immediately preceding portion of the format may be repeated one or more times.

COMMENTS IN SORT DESCRIPTION

Any string of characters that is preceded by and terminated by a slash (/) will be treated as a comment in the Sort Description. Comments cannot be inserted within a word of the Sort Description. If an incomplete comment is detected, an error message is issued and the Sort is terminated.

FILES STATEMENT (FOR SORT)

The FILES statement specifies the files to be processed by Sort. The pathnames specified in the FILES statement can be full pathnames or relative pathnames related to the current working directory.

The format of the FILES statement is:

FILES:-IF path,-OF path[,-WF path];

-IF path

Specifies the pathname of the file containing input records to be sorted. Required argument.

-OF path

Specifies the pathname of the output file. The output file, if on a disk, must be created prior to execution of the Sort. If the output file is to be on tape, then the characters must be defined through the argument of the GET command. Required argument.

-WF path

Defines the file to be used as a permanent work file by Sort. The work file must be a disk file created before Sort is executed. If this argument is not specified, Sort creates a temporary work file within the current working directory; the file is deleted when Sort terminates. Unless the file to be sorted is small, for performance reasons a permanent work file is recommended.
Optional argument.

Example 1:

The file INFILE contains records to be sorted; the file OUTREC is the file on which sorted records are to be written. These two files are on the same volume and are identified by relative pathnames related to the current working directory. The work file used by Sort is WORK2, which resides on the volume SRTWK. The FILES statement is:

FILES:-IF INFILE,-OF OUTREC,-WF ^SRTWK>WORK2;

Example 2:

The single file ITSV23, containing input records, resides on the tape volume SRTI64; the tape is mounted on unit number 00. The output file OUTLIB resides on the disk volume Z10054. The Sort work file SWK01 is on the volume Z10193. The FILES statement is:

```
FILES: -IF  >SPD>MT900>SRT164>ITSV23
-OF  ^Z10054>OUTLIB,-WF  ^Z10193>SWK01;
```

Note:

MOD 200 users must replace the ">SPD>" argument immediately preceding device specifications with the tilde (~). Also, refer to the MOD 200 *Application Development Guide* for MOD 200 specific symbolic device names.

INCL AND OMIT STATEMENTS (FOR SORT)

The INCL and OMIT statements cause input file records to be processed by Sort only if they meet certain condition(s). You can designate the criteria to be used by Sort for determining which record(s) *will* or *will not* be processed by specifying the INCL or OMIT statement, respectively. If INCL or OMIT is not specified, all input file records are processed.

Depending on which arguments are entered, two fields within a record are compared or a field is compared to a specified value. Within a single execution of Sort, either one to four INCL statements or one to four OMIT statements may be specified. Within a single sort, a maximum of four conditions may be specified; they may be within a single statement or be interspersed among up to four statements.

The order in which INCL or OMIT statements are specified determines the order in which records are tested. If a record meets all of the conditions specified in a single statement, it is not tested against the condition(s) specified in any subsequent statements. If the condition(s) are met, the specified record is processed (if INCL was used) or not processed (if OMIT) was used.

The format of the INCL and OMIT statements is:

$$\left\{\begin{matrix}\text{INCL}\\ \text{OMIT}\end{matrix}\right\} \text{:criteria\_description}\left[\ \text{AND}\ \ \text{criteria\_description}\ \ldots\right];$$

The format of the criteria description is determined by the type of comparison being specified:

Comparing two fields within a record:

data_type(size),position $_1$,operator,position $_2$

Comparing a field to a specified value:

data_type(size),position $_1$, operator,'literal'

data_type

Data type of the key field is specified by a predefined code. See Table 2-1 later in this section for data types that can be used, and the code associated with each type. Required argument.

size

Size of the key field expressed as a decimal integer. The size is the number of data type units that constitute the field, excluding the unit that contains a separate sign. The units may be bit, four-bit, or byte elements. Any word separators, including parentheses, can be used with the integer representing size. (In the above format, parentheses have been inserted for readability.) Required argument.

position $_1$

Position of the beginning byte of the key field to which another key field or a literal value is being compared; the position is relative to the beginning of the record. Expressed as a decimal integer. The record is considered to be aligned on a byte boundary, with the first byte of the record being numbered 1. Required argument.

operator

Type of comparison that will be made between two specified fields or a specified field and a specified value; must be one of the following:

EQ - Equals
NE - Is not equal to
LT - Is less than
LE - Is less than or equal to
GT - Is greater than
GE - Is greater than or equal to

Required argument.

position $_2$

Position of the beginning byte of the key field that is being compared to the key field designated in position $_1$. The position is relative to the beginning of the record and is expressed as a decimal integer. The record is considered to be aligned on a byte boundary, with the first byte of the record being numbered 1. Optional argument.

literal

Alphanumeric character string constituting the value to which the key field designated in position $_1$ will be compared; must be delimited by single quotes. If the literal is a positive numeric value, a plus sign (+) need not be used.

Note:

A literal may span more than one line. If a literal begins at the beginning of a line or part of a literal is at the beginning of a line, that line may not begin with the characters QT, QUIT, or !S.

If a literal is to be compared to a character string, the literal must comprise the same number of characters as that field, including trailing blanks. (The number of characters is designated in the size argument.)

Note:

A single quote may be included as part of the character string literal by using two contiguous single quotes wherever a single quote is desired. For example, the literal DEPT. 'AA' is written 'DEPT. ''AA'''.

If a literal is to be compared to a field that is a numeric data type, the literal is converted to the appropriate numeric data type. If the literal is shorter than the size specified in the size argument, the converted literal is right-justified with leading zeros; if the literal is longer than the size specified, it is an error condition.

Within a single Sort Description, a maximum of 128 characters may be entered for all of the specified literals. Optional argument.

Example 1:

Statement with One Condition:

This statement requests that a character string (CHAR) comprising 5 units ((5)) that begins on the fourth byte of the record (4) be compared to the value ABCDE ('ABCDE'). If they are equal (EQ), the record *will* be processed (INCL) by Sort.

```
INCL:CHAR(5),4,EQ,'ABCDE';
```

Note that in this example commas are used as delimiters whereas in Example 2 spaces are used as delimiters.

Example 2:

Statement in which Two Conditions Must be Met:

In this statement, both of the specified conditions must be met (AND); otherwise, the record is not processed by Sort. A signed binary field (BIN) comprising 15 units ((15)) that begins on the fourth byte of the record (4) must be less than (LT) a field with the same attributes that begins on the tenth byte (10). The value of a signed decimal overpunch field (DEC) must be equal to (EQ) the value +26 ('26').

```
INCL:BIN(15) 4 LT 10 AND DEC(2) 15 EQ '26';
```

Note the mandatory use of single quote delimiters for the numeric literal.

Example 3:

Statements in which One or Two Conditions Must be Met:

Since the following conditions are *not* within a single statement separated by AND, the record is not processed (OMIT) if either of the specified conditions is met.

The first statement designates that a signed, packed decimal field (PDEC) comprising five units ((5)) that begins on the third byte of the record (3) be compared to the value -12345; if the PDEC field is greater than (GT) -12345, the record is *not* processed (OMIT) by Sort. If this condition is met, the second statement is not processed; otherwise, a signed decimal field with a trailing but separate sign (TDEC) comprising four units ((4)) that begins on the tenth byte of the record (10) is compared to the value 1024. If the TDEC field is less than (LT) 1024, the record is omitted (OMIT) from Sort processing.

```
OMIT:PDEC(5),3,GT,'-12345';
OMIT:TDEC(4),10,LT,'1024';
```

KEYS STATEMENT (FOR SORT)

The KEYS statement defines key fields in the input records to be used in sequencing the records in the output file. The statement consists of a series of 1 to 16 key descriptions, each of which specifies a single key field. The key descriptions must be specified in order, the first describing the major key and the last describing the least significant minor key. Key fields can be overlapping. In files with variable length records, the smallest record to be sorted must contain all the key fields. A record that does not contain all key fields will not be included in the sort. Each key description indicates whether the values for that key field are to be ranked in ascending or descending order.

Note:

If both KEYS and ARRange are specified, the key field(s) must be contained within the specified ARRange field (see "ARRange Statement (For Sort)" below).

The format of the KEYS statement is:

KEY[S]:key_description[,key_description . . .];

Each key description has the following format:

data_type(size),position[,D]

data_type

Data type of the key field is specified by a predefined code. See Table 2-1 for data types that can be used, and the code associated with each type. Required argument.

size

Size of the key field expressed as a decimal integer. The size is the number of data type units that constitute the field, excluding the unit that contains a separate sign. The units may be bit, four-bit or byte elements.

Any word separators, including parentheses, can be used with the integer representing size. (In the above format, parentheses have been inserted for readability.) Required argument.

position

Position of the beginning byte of the specified key field, relative to the beginning of the record. Expressed as a decimal integer. The record is considered to be aligned on a byte boundary, with the first byte of the record being numbered 1. Required argument.

D

Descending order of values of the specified key field to be used in ranking the records. Optional argument. Default value is ascending order.

Section 3

Reports and Messages Issued by Sort

Sort delivers a Sort Report to the user-out file on completion of the sorting process. In addition, Sort issues error messages, as appropriate, to the error-out file. The Sort Report and error messages are discussed in this section.

SORT REPORT

The Sort program always delivers a Sort Report to the user-out file unless Sort is terminated because of parameter diagnostic errors. The contents of the Sort Report are determined by the control arguments specified in the invoking SORT command.

BASIC SORT REPORT

If no control argument related to the contents of the Sort Report appears in the SORT command, the report issued by Sort is always a basic Sort Report. The basic Sort Report identifies the input and output files, provides statistical information on the records processed, and indicates whether error messages were issued during execution of the sort application. The items in the basic Sort Report (explained in Table 3-1 later in this section) are as follows:

```
SORT -version number -date/time linked
INPUT FILE: pathname
RECORDS READ nnnnnn
OUTPUT FILE: pathname
RECORDS WRITTEN nnnnnn
RECORDS DELETED nnnnnn
FATAL ERROR IN SORT          (Conditional message)
WARNING ERROR IN SORT        (Conditional message)
```

All items of the basic Sort Report are supplied whenever the Sort Report is printed, whether or not additional items are included.

EXPANDED SORT REPORT (INCLUDING SORT DESCRIPTION)

The optional control argument -PD can be specified in the invoking SORT command to extend the contents of the Sort Report. The argument -PD means "print Sort Description." If -PD is specified, the complete Sort Description as originally submitted is printed at the beginning of the Sort Report, followed by the other items of the Sort Report.

Table 3-1 illustrates the format of the expanded Sort Report which is delivered to the user-out file if the -PD option was specified in the SORT command. The pathnames cited in Table 3-1 are full pathnames, obtained after the files to which they refer are opened.

TABLE 3-1. FORMAT OF SORT REPORT (INCLUDING BASIC SORT REPORT AND SORT DESCRIPTION)

Sort Report Item	Comment
****SORT DESCRIPTION**** Sort Description statements	Each line of the Sort Description is entered exactly as originally submitted to Sort. (Included only if -PD specified in SORT command.)
Sort -version number -date/time linked	Header for main body of Sort Report. (Always included; part of basic Sort Report.)
INPUT FILE: pathname	Full pathname of input file as returned by the system. (Always included; part of basic Sort Report.)
RECORDS READ nnnnnn	Number of data records read from the Sort input file. (Always included; part of basic Sort Report.)
OUTPUT FILE: pathname	Full pathname of Sort output file as returned by the system. (Always included; part of basic Sort Report.)
RECORDS WRITTEN nnnnnn	Number of data records written to the output file. (Always included; part of basic Sort Report.)
RECORDS DELETED nnnnnn	Number of records deleted by the Sort program when the -DL argument is specified or due to record selection based on INCL or OMIT statements. (Always included; part of basic Sort Report.)
FATAL ERROR IN SORT	Written to the user-out file *only* if a fatal error message has been sent to the error-out file. (When issued, considered part of basic Sort Report.)
WARNING ERROR IN SORT	Written to the user-out file *only* if one or more warning error messages have been sent to the error-out file. This message is issued if the RECORD TOO SMALL error message was previously generated. (When issued, considered part of basic Sort Report.)

Example

A typical Sort Report delivered to the user-out file is shown below. The invoking SORT command for this sort application contains the control argument -PD.

```
**** SORT DESCRIPTION ****
FILES:-IF IOFILE -OF OUFILE -WF WKFILE;
KEYS:CHAR(6) 13 ;
SORT -version number -date/time linked
INPUT FILE:    ^ SORTVL>IOFILE
RECORDS READ 001000
OUTPUT FILE:    ^ SORTVL>OUFILE
RECORDS WRITTEN 001000
RECORDS DELETED 000000
```

ERROR MESSAGES ISSUED BY SORT

A complete list of error messages generated by Sort and delivered to the error-out file appears in the *System Messages* manual and in the MOD 200 *Operator's Guide.*

As soon as the Sort program is invoked, it begins to scan all elements of the SORT command and the Sort Description. If it detects an error, it issues an error message describing that error. Sort scans the entire Sort Description and issues error messages as appropriate. Sort then terminates if it detected one or more errors in the Sort Description.

For syntax errors, Sort generates an error message consisting of a coded number and the words PARAMETER SYNTAX ERROR. This message is followed by a secondary message that either identifies the error or presents a string of characters that can be interpreted as follows:

- First word of string indicates location of error
- Remaining words of string indicate words that Sort is unable to scan intelligently. All word separators are reduced to blanks.

Example:

The following KEYS statement is submitted:

```
KEYS:CHAR(A),4,CHAR(3),10,CHA(4),15;
```

The error messages Sort issues as a result of scanning this line are:

```
313121 PARAMETER SYNTAX ERROR
3131FF "A 4"
313121 PARAMETER SYNTAX ERROR
3131FF "CHA 4 15"
```

Section 4
Merge Language

The Merge program is invoked through specification of the MERGE command. The MERGE command provides information to specialize the Merge program for a particular execution, including identification of the file containing the Merge Description.

The Merge Description contains additional information for specializing the Merge, including specification of:

- The input and output files to be used by Merge
- One or more key fields to be used for merging records
- The criteria to be used to determine which records of the Merge input file(s) will be merged
- The record arrangement of the output file

The MERGE command and the statements that constitute the Merge Description are described in detail below. This section also includes examples of typical Merge usage.

MERGE COMMAND

The MERGE command invokes the Merge program. Any pathname specified in the MERGE command can be either a full pathname or a relative pathname related to the current working directory. (Refer to the appropriate *System Concepts* manual for a description of the use of pathnames in commands or to the MOD 200 *Operators Guide*.)

The format of the MERGE command is:

MERGE [ctl_arg]

[ctl_arg]

Control arguments. Any or all of the following optional control arguments can be used:

-IN path

Specifies the name of the file containing the Merge Description for this merge. If not specified, the user-in file is used.

-PD

Indicates that a listing of the Merge Description is to be produced on the user-out file. (Only the first 71 characters of each line will be displayed.)

-DL

Designates that if Merge encounters records that have the same key, only the first record defined by the Merge FIFO rule is written to the output file (see the -IF path argument of "FILES Statement (For Merge)" later in this section).

Example 1:

The MERGE command and Merge Description are to be submitted in the command-in file. The Merge program will be found according to your search rules (see the LIST SEARCH RULES (LSR) command in the *Commands* manual, MOD 400 and MOD 600 only). The invoking command is:

```
MERGE
```

Example 2:

The Merge program is stored on the volume MYAPPL; the Merge Description is identified by the relative pathname MD. If Merge encounters records that have the same key, only one of the records is written to the output file; the others are deleted. The invoking command is:

```
^MYAPPL>MERGE -IN MD -DL
```

MERGE DESCRIPTION

The Merge Description consists of the following statements, which supply information to specialize the Merge for a particular application:

- FILES statement: (Required) Specifies the input and output files for the Merge application.
- INCL/OMIT statement(s): (Optional) Specify which records of the Merge input file(s) will be processed.
- KEYS statement: (Required) Describes the input file fields to be used for merging files.
- ARRange statement: (Optional) Designates the placement of input record byte strings within the output record.

The Merge Description is required. The FILES statement must be first. If INCL or OMIT statements are used, they must precede the KEYS and optional ARRange statements. The syntax of the Merge Description is the same as that for the Sort Description (see "Syntax of Sort Description" and "Comments in Sort Description" in Section 2).

FILES STATEMENT (FOR MERGE)

The FILES statement specifies the pathnames of the input file(s) to be processed by Merge, and the output file. There may be one through six input files. Pathnames specified in the FILES statement can be full pathnames or relative pathnames related to the current working directory.

The format of the FILES statement is:

FILES:-IF path[,path . . .],-OF path;

-IF path

Specifies the pathname(s) of the input file(s) to be processed by Merge. Up to six input file pathnames may be specified. If duplicate records are found (i.e., they have the same key field), the records are written to the output file in the order in which the files containing them appear in the input file list. Required argument.

-OF path

Specifies the pathname of the output file. The output file, if on a disk, must be created prior to execution of the Sort. If the output file is to be on tape, then the characters must be defined through the argument of the GET command. Required argument.

Example:

Three files to be merged reside on different volumes; the first file is relative to the current working directory. The output file resides on the same volume as the input file. The FILES statement is:

```
FILES:-IF BRANCH1,  ^DIST01>BRANCH2,
 ^DIST02>BRANCH3,-OF SUMM;
```

INCL AND OMIT STATEMENTS (FOR MERGE)

The INCL and OMIT statements cause input file records to be processed by Merge only if they meet certain condition(s). You can designate the criteria to be used by Merge for determining which record(s) will or will not be processed by specifying the INCL or OMIT statement, respectively. For a more detailed description of these statements, see "INCL and OMIT Statements (For Sort)" in Section 2.

The format of the INCL and OMIT statements is:

$$\left\{\begin{matrix}\text{INCL}\\\text{OMIT}\end{matrix}\right\}\text{:criteria\_description}\left[\ \text{AND criteria\_description . . .}\ \right];$$

ERROR MESSAGES ISSUED BY MERGE

Merge scans the MERGE command and the Merge Description, as they are entered, and issues appropriate error messages, if any. If one or more errors are detected in the Merge Description, execution of Merge terminates.

For syntax errors, Merge generates an error message comprising a coded number and the words PARAMETER SYNTAX ERROR. For a description of parameter syntax errors, see "Error Messages Issued by Sort" in Section 3.

A complete list of error messages generated by Merge and delivered to the error-out file appears in the *System Messages* manual and in the MOD 200 *Operator's Guide.*

Section 6

Operating Procedures for Sort and Merge

This section describes the procedures for creating disk files for Sort and Merge, and preparing and submitting the commands and statements that invoke and particularize Sort and Merge for specific applications. Also included in this section are sample sort and merge runs.

PREPARING FILES FOR SORT AND MERGE

The Sort and Merge output files, if on a disk, must have been created as files (on a storage module, cartridge disk, or diskette) before Sort or Merge execution, respectively. If a permanent file is to be used as the Sort work file, it too must be created before execution of Sort. The CREATE FILE (CF) command (described in the *Commands* manual and in the MOD 200 *Operator's Guide*) is normally used to create a disk file. Preparation and disposition of the work and output files are described below.

SORT WORK FILE

Sort requires a single-volume disk work file. To meet Sort requirements, it is recommended that the following be specified in the command that creates the work file:

CF path -SIZE n

The formula for determining the initial allocation size n of the file is as follows:

$$n = 1.2 * \frac{\text{(size of output file in sectors)}}{s}$$

where s equals 4 for diskette or 2 for other disk devices. The value of n is rounded up.

During Sort execution, if the initial work file size allocation n is insufficient for the size of the file to be sorted, an attempt is made to expand the work file (see the -INC_SIZE argument of the CREATE FILE command in the *Commands* manual and in the MOD 200 *Operator's Guide*). If the file cannot be incremented, Sort is terminated.

Example:

The output file will occupy 822 sectors on a cartridge disk. The control interval size is 512 bytes. The size of the work file is calculated as follows:

$$n = 1.2 * \frac{822}{2}$$

493.2 or 494 rounded up

The command to create the work file is:

CF ^VLSORT>WKSRT -SIZE 494

The work file, once created, can be considered a permanent work file. It can be used for any sort application for which it has sufficient capacity. However, it cannot be used by two Sort programs concurrently.

SORT AND MERGE OUTPUT FILES

Your requirements determine the characteristics of the Sort and Merge output files. The attributes of the output file may differ from those of the input file(s). The only restriction on output file organization is that it cannot be indexed. If the sequential file specified for output already contains records prior to execution of the Sort/Merge, these records are overwritten by Sort/Merge output. The output file must be created prior to the execution of Sort/Merge.

If the output records have a predefined size that differs from the size of the input records, the following occurs when the output record is written:

Record Size	Results
input<output	The input record contents are left justified; the remainder of the output record is set to spaces unless the output record type is variable.
input>output	The input record contents are left justified; truncation occurs as required.

DISPOSITION OF FILES

The shareability attribute of the work file is exclusive-use only. While the input and output files' shareability attribute is defaulted to exclusive-use only, this can be overridden by means of the GET command which is described in the *Commands* manual or MOD 200 *Operator's Guide*. Because input and output buffers are drawn from either the user's or the system's memory, depending on the shareability attribute of the respective files, the user normally assigns the same attribute to both files. If the attribute is exclusive-use only, buffers are drawn from the user's memory; otherwise, system memory is used.

The input file can be rewritten as the output file of the Sort.

The Sort input, output, and work files can reside on the same volume; however, for best performance, it is advisable to have the work file reside on a separate volume whenever possible. If the input file and output file do not reside on the same volume, the output volume need not be mounted when Sort is invoked; mounting can be deferred until the output file is needed. When the volume containing the input file is being used exclusively by Sort, the volume containing the output file can be mounted on the device from which the input volume has been demounted.

The Merge input and output files can reside on the same volume(s). All volumes must be simultaneously mounted.

SORT AND MERGE LOGICAL FILE NUMBERS

Logical file numbers (LFNs) for the Sort input, output, and work files are dynamically obtained from the system; similarly, LFNs for the Merge input and output files are dynamically obtained from the system. Sort and Merge do not use reserved LFNs. Sort uses two user-available LFNs, whereas Merge requires two through seven user-available LFNs.

CREATING THE SORT AND MERGE DESCRIPTIONS

The Sort and Merge Descriptions can be entered interactively or stored in the user-in file.

The Editor can be used to create a Sort/Merge Description file on a disk volume. The name of the Description file is specified in the Editor write directive (W). When the Editor processes the quit directive (Q), it supplies an end-of-file indicator to the Description file. (For a description of the Editor, see the *Program Preparation* manual. Information on using the Editor appears in the *Programmer's Guide* and in the MOD 200 *Application Development Guide*.)

Termination of the Description must be indicated by specification of the single word QUIT (or QT) on a separate line, unless an end-of-file indicator appears at the end of the Description file.

INVOKING THE SORT AND MERGE PROGRAMS

To invoke the Sort or Merge program via the command processor, enter the SORT or MERGE command, respectively. The principal means of submitting these commands are:

1. Invoking Sort or Merge interactively (MOD 400 and MOD 600 only): The SORT and MERGE commands can be submitted by hand through a terminal. During activation of the online task group under which Sort or Merge is to be executed, the terminal must be specified as the file from which commands and your input are to be read. (See the *Commands* manual for a description of the ENTER GROUP REQUEST command, which causes an online task group to be activated.)

2. Invoking Sort or Merge from an EC file: The SORT or MERGE command can be stored in a file created by the Editor and identified by the pathname path.EC. When the pathname of the EC file is specified in an EXECUTION COMMAND (EC), the command is read and passed to the command processor for execution. (The EXECUTION COMMAND is described in the *Commands* manual and MOD 200 *Operator's Guide*.)
3. Invoking Sort or Merge from a batch command file (MOD 400 and MOD 600 only): The SORT or MERGE command can be stored in a file from which the command processor reads its commands during execution of a batch task group. The pathname of the file must be specified in the in_path argument of the ENTER BATCH REQUEST (EBR) command that causes the batch task group to be executed. (The ENTER BATCH REQUEST command is defined in the *Commands* manual.)

SUBMITTING THE SORT AND MERGE DESCRIPTIONS

SPECIFYING SORT OR MERGE DESCRIPTION FILE IN -IN ARGUMENT OF SORT/MERGE COMMAND

The invoking SORT or MERGE command designates the file from which the Sort or Merge program will read the Sort or Merge Description, respectively. If the -IN path argument is specified in the SORT/MERGE command, the pathname in that argument identifies the file containing the Description.

SUBMITTING SORT OR MERGE DESCRIPTION WHEN -IN ARGUMENT OMITTED FROM SORT/MERGE COMMAND

If the invoking SORT/MERGE command does not include an -IN argument, the Sort/Merge Description is read from the file currently being used as the user-in file. The location of the current user-in file is related to the means used to submit the SORT/MERGE command (see "Invoking the Sort and Merge Programs" above). Listed below are various means by which the SORT and MERGE commands can be submitted to the system and the location of the current user-in file in each case.

1. SORT or MERGE command entered interactively: If the SORT or MERGE command without the -IN argument is entered by hand through a terminal, normally the Sort/Merge Description is also entered interactively through the terminal. (A description of the procedure for entering the Description interactively is given later in this section.)
2. SORT or MERGE command in EC file: If the SORT or MERGE command without the -IN argument is stored in and read from an EC file, the related description can be located in one of two places:
   a. In the EC file, following the SORT/MERGE command. The EC control directive &A must appear in the EC file, on a separate line preceding the SORT/MERGE command, so that the EC file can be substituted for the user-in file.
   b. In the current user-in file (in the online dimension, normally the user's terminal). The EC file must not contain the EC control directive &A on a line preceding the SORT/MERGE command.
3. SORT or MERGE command in a batch command file: If the SORT or MERGE command without the -IN argument is stored in and read from a file whose pathname is specified in the in_path argument of the ENTER BATCH REQUEST (EBR) command, normally the Description is also stored in and read from that file.

Whenever the -IN argument is omitted from the SORT/MERGE command line, the message

ENTER { SORT / MERGE } DESCRIPTION

is written to the user's terminal. In case 1, above, the user responds to this message by entering the SORT/MERGE description through the terminal (see below). In case 2 and 3, above, the SORT/MERGE description is read automatically with no further user action required.

PROCEDURE FOR ENTERING SORT OR MERGE DESCRIPTION INTERACTIVELY

To enter the Sort or Merge Description through a user's terminal, perform the following steps:

1. After being invoked, the Sort and Merge programs issue the following message on the terminal:

   ENTER $\left\{\begin{matrix}\text{SORT}\\\text{MERGE}\end{matrix}\right\}$ DESCRIPTION

   In response, enter the Sort or Merge Description manually, line by line.

2. At the completion of the Description, enter the characters QUIT (or QT) at the beginning of a separate line, to indicate termination of the Description.

3. As the Sort or Merge program processes each line of the Description, it issues an error message to the error-out file for each error in syntax or number of parameters that it detects. The program continues to accept Description lines until it encounters the word QUIT (or QT). If the program detects no errors, it continues to execute. Otherwise, it returns to the command level.

4. If Sort or Merge terminates because of a fatal error, you can revise and resubmit the SORT or MERGE command and Sort/Merge Description.

5. Should an error occur during typing of the Description on the terminal, you can terminate Sort/Merge by typing !SΔ as the first characters on the next line. Once the program reads these characters, it suppresses subsequent error messages, causes the hexadecimal value FFFF to be supplied to register R1, and returns control to the command level.

6. When Sort or Merge is executing from an EC file and !SΔ is entered through the terminal to terminate the program, control is returned to the EC file. Normally, the next command is automatically read from that file. However, under certain conditions, control can be transferred from the EC file before the next command is read. For an exit from the EC file to occur before the end of the EC file, the EC control directive &IFΔ must exist in the EC file on the line immediately following the SORT or MERGE command. The following complete &IFΔ control directive must have been stored on that line:

   &IFΔ[EQUALSΔ&STATUSΔO]Δ&THENΔ&ELSEΔ&QUIT

   This directive causes the error status code in register R1 to be interrogated. The above control statement is interpreted as follows:

   > If the error status code is zero (0), continue with the next command or directive line; otherwise, quit.

   Because specifying !SΔ causes the error status code to contain a nonzero value, EC file execution is terminated.

SAMPLE SORT RUNS

The examples given below illustrate the input streams required to cause two sort runs to be executed. Included in each example is the resulting Sort Report.

Example 1:

The SORT command and the Sort Description are both to be entered through an interactive terminal. The amount of memory requested to support Sort execution is 16K words.

Files used by Sort include:

1. An input file on volume DSK011 referred to by the pathname ^ DSK011>AC>PAY.
2. An output file on the same volume referred to by a pathname ^ DSK011>TAXD> WAGE.
3. The Sort work file SRTWK on volume WDIR01.

Two record keys are to be used in sorting: a 10-character field beginning in the seventh byte of the record, and a two-character field beginning in the third byte. The latter key is to be sequenced in descending order.

The following terminal typeout shows the interactive dialog required to invoke and run the Sort program, and the resulting Sort Report. Entries supplied by the user are labeled (u)

and those supplied by the system are labeled (s) in the margin to the left of the listing. (These labels do not appear on an actual listing.)

```
(u) SORT -SIZE 16                                    (Invoking SORT command)
(s) ENTER SORT DESCRIPTION                           (System request for Sort Description statement)
(u) FILES:-IF ^DSK011>AC>PAY                         (Sort Description statement)
    -OF ^DSK011>TAXD>WAGE -WF ^WDIR01>SRTWK;
(u) KEYS:CHAR(10),7,CHAR(2),3,D;                     (Sort Description statement)
(u) QUIT                                             (Terminate Sort Description)
(s) SORT -version number -date/time linked           (Sort Report header)
(s) INPUT FILE: ^DSK011>AC>PAY                       (Item of basic Sort Report)
(s) RECORDS READ: 2000                               (Item of basic Sort Report)
(s) OUTPUT FILE: ^DSK011>TAXD>WAGE                   (Item of basic Sort Report)
(s) RECORDS WRITTEN: 2000                            (Item of basic Sort Report)
(s) RECORDS DELETED: 0                               (Item of basic Sort Report)
```

Example 2:

In this example, the Editor is used to create a file (SRTDES) containing the Sort Description under the current working directory. Commands to invoke the Editor and Sort are submitted through a terminal. The resulting Sort Report, which includes a listing of the Sort Description statements, is to be printed on the terminal (designated as the user-out file).

The Sort work file (WKFILE) is under the current working directory. The input file (RECIN) and output file (RECOUT) are on the same volume (VOL234). Keys to be used in ranking the input records are as follows (the first byte of the record being numbered 1):

1. A 12-character field starting in byte 3
2. A 31-character field starting in byte 20 (to be sorted in descending order)
3. A five-character field starting in byte 6

The amount of memory to be allocated for Sort execution is 20K words.

The Editor directives required to create the Sort Description file are listed below, followed by the invoking SORT command, and the resulting Sort Report. (Refer to the *Program Preparation* manual or to the MOD 200 *Application Development Guide* for detailed information on the Editor.) The originator of each of the following entries is indicated in the left margin: (u) for user; (s) for system. (These labels do not appear on an actual listing.)

```
(u) ED                                               (Invoking the Editor)
(u) I                                                (Editor insert directive)
(u) /SRTDES/                                         (Comment line giving file name)
(u) FILES:-IF ^VOL234>RECIN                          (Sort Description statement)
    -OF ^VOL234>RECOUT -WF WKFILE;
(u) KEYS:CHAR(12)3,                                  (Sort Description statement)
    CHAR(31)20 D,CHAR(5)6;
(u) !F                                               (Terminate insert and enter Edit mode)
(u) W SRTDES                                         (Editor write directive naming the Sort Description file)
(u) Q                                                (Quit; exit from Editor)
(u) SORT -IN SRTDES -SZ 20 -PD                       (Invoking SORT command)
(s) ****SORT DESCRIPTION****                         (Initial entry of Sort Report)
(s) /SRTDES/
(s) FILES:-IF ^VOL234>RECIN                          }
    -OF ^VOL234>RECOUT -WF WKFILE;                   } (Listing of Sort Description statements in Sort Report)
(s) KEYS:CHAR(12)3,                                  }
    CHAR(31)20D,CHAR(5)6;                            }
(s) SORT -version number -date/time linked           (Sort Report header)
(s) INPUT FILE: ^VOL234>RECIN                        (Item of basic Sort Report)
(s) RECORDS READ: 500                                (Item of basic Sort Report)
(s) OUTPUT FILE: ^VOL234>RECOUT                      (Item of basic Sort Report)
(s) RECORDS WRITTEN: 500                             (Item of basic Sort Report)
(s) RECORDS DELETED: 0                               (Item of basic Sort Report)
```

SAMPLE MERGE RUN

Example:

The MERGE command and the Merge Description both are to be entered through an interactive terminal. Files used by Merge include:

1. Input file on volume ZSVOLC referred to by the pathname ^ZSVOLC>APPLE.
2. Second input file on volume VL6469 referred to by the pathname ^VL6469>PEAR.
3. Output file on volume ZSVOLC referred to by the pathname ^ZSVOLC>PLUM.

The following single key is used to merge the two input files: a character string comprising six bytes starting on the first byte of the record.

The following terminal typeout shows the interactive dialog required to invoke and run the Merge program, and the resulting Merge Report.

```
MERGE
ENTER MERGE DESCRIPTION
FILES: -IF ^ZSVOLC>APPLE ^VL6469>PEAR -OF ^ZSVOLC>PLUM;
KEYS: CHAR 6 1;
QT
MERGE -version number -date/time linked
INPUT FILE 1: ^ZSVOLC>APPLE
INPUT FILE 2: ^VL6469>PEAR
OUTPUT FILE: ^ZSVOLC>PLUM
RECORDS READ 01:000241
RECORDS READ 02:000137
TOTAL READ: 000378
RECORDS WRITTEN 000378
RECORDS DELETED 000000
```

*

Section 7

Sorting Using Subroutine Calls (MOD 400 and MOD 600 Only)

An application program may use a sequence of subroutine calls to sort records. To use these calls, link an interface module to your program, which may be written in COBOL, FORTRAN, or GCOS 6 assembly language. The sorting functions are performed by a dynamically loaded bound unit, SORTC. A call passes each record, and these records are sorted using a disk work file. After the calling program is notified that the sort has completed, call for the return of each record in sorted order. Prior to passing records to be sorted, call for the initialization of the sort by specifying the maximum amount of memory to be used, record and work file attributes, and the sort key fields. The disk work file may be temporary or permanent. The minimum memory requirement for Sort is 8K words.

Key fields have the following characteristics:

- Up to 16 key fields may be specified; they may be contiguous, separated, or overlapped.
- Keys may be sorted in ascending or descending order according to the ASCII collating sequence.
- Key field data types may be any of the following:
  - Character string
  - Single- or double-word signed binary
  - Unpacked decimal: unsigned or signed with the sign leading, trailing, or trailing over-punched
  - Packed decimal: unsigned or signed trailing

One major and up to 15 minor keys may be specified. Records are ordered first according to the major key; then all records containing the same major key are sorted according to the minor keys, in the sequence dictated by the sort description, which is described later in this section.

SORT SUBROUTINE CALLS

The function of each call is described briefly below; following are more detailed descriptions. Normally each function is required, except for abort sort. At the completion of each call, the subroutine is closed; i.e., control returns inline to the calling program immediately following the call.

- Initialize sort (ZSSRT) — Establishes the environment, according to the sort description, for the sorting process.
- Release record (ZSREL or ZSRELD) — Passes a record to Sort.
- Commence sort (ZSCOMM) — Performs the sorting process.
- Return record (ZSRET or ZSRETD) — Returns the next sorted record to the calling program.
- Abort sort (ZSEND) — Terminates execution of Sort prematurely, provided all records have not been returned.

SUBROUTINE CALL ARGUMENTS

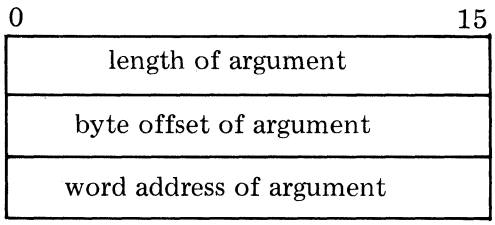
DOPE VECTORS

Each subroutine call is associated with an argument list. Some arguments are passed using dope vectors. In the detailed descriptions of subroutine calls, it is indicated for each argument whether a dope vector is used.

The higher level programmer must be aware that the only types of data declarations that cause a dope vector to be used are:

- COBOL — An argument declared as elementary DISPLAY. (In COBOL it may be necessary to use an 01 level elementary REDEFINES to a group item that is to be referenced with a dope vector.)
- FORTRAN — An argument declared as CHARACTER.

If the application program using the sort subroutine calls is written in assembly language, the GCOS 6 standard calling sequence must be followed; i.e., arguments are passed through an argument list referenced by register B7. Depending on the form of data that is passed, the address pointers in the argument list either point directly to the data or to a dope vector, which in turn points to the data. The main intent of dope vectors is to permit data items that are not aligned on word boundaries to be passed. A dope vector consists of three contiguous words in the following format:



The initialize Sort call requires that the sort description argument be passed with a dope vector. Consequently, the assembly language name used for the sort description argument must reference a dope vector that in turn points to the sort description. It is the assembly language programmer's responsibility to build the dope vector.

ARGUMENTS COMMON TO SORT SUBROUTINE CALLS

The first two arguments are the same for each of the five calls. The first argument specifies a work area within your program, and the second argument specifies the field used for storing the return code at the completion of a call. Both arguments are described in detail below.

work area (work_area)

This argument specifies the name of the work area declared within your program. During the execution of Sort, the work area must be available for the exclusive use of Sort. The work area must be word-aligned, and must occupy at least 21 words. It is *not* passed using a dope vector.

Throughout the sort, the work area retains the sort's context; therefore, the same argument value must be used for all calls associated with a specific sort. When a call is made to initialize the sort, the first word of the work area must be the binary equivalent of 4096. It is changed during the sort process, but it is reset to 4096 when the sort terminates.

report status (return_code)

This argument is the name of a 16-bit binary integer return code field used by Sort to report status to the caller. Unlike the work area argument, it is not necessary to specify the same argument value in all calls made during a sort. If the application program is written in COBOL, the status code should be declared as COMPUTATIONAL-1. If the program is written in FORTRAN, the status code should be declared as INTEGER. This argument is *not* passed using a dope vector.

ZSRET

Indicates that the record area (specified in the record_ar argument) is to be passed *without* a dope vector.

ZSRETD

Indicates that the record area (specified in the record_ar argument) is to be passed *with* a dope vector.

work_area

Name of the work area used in the ZSSRT call. Required argument.

return_code

Name of the 16-bit binary integer return code field used by Sort to report status and errors. On a normal return of the last record, the value of the return code is the binary equivalence of 5. This argument must *not* be passed with a dope vector. Required argument.

record_ar

Name of the area into which the record will be returned. The record may be released to and returned to the same area (i.e., the name specified in this argument may be the same as the name specified in the record_id argument of the release record call). If your declaration of this area requires that the argument be passed using a dope vector, then the call ZSRETD must be used; otherwise, the call ZSRET must be used. Required argument.

record_length

Length of the record, in bytes, returned by Sort. The size of the record length field is one word. In COBOL, the record length is declared COMP-1. In FORTRAN, the record length is declared INTEGER. Required argument.

ABORT SORT CALL

The abort sort function may be used to terminate execution of Sort before all of the records have been returned. The system will perform functions normally done after the last record is returned; i.e., the work file is closed (and released if it is a temporary file), and the memory used by Sort is released.

The formats for the abort sort call are:

COBOL

CALL "ZSEND" USING work_area, return_code

FORTRAN

CALL ZSEND (work_area, return_code)

Assembly Language

CALL ZSEND,work_area,return_code

work_area

Name of work area used in the ZSSRT call. Required argument.

return_code

Name of the 16-bit binary integer return code field used by Sort to report status and errors. This argument must *not* be passed with a dope vector. Required argument.

SEQUENCING OF SORT SUBROUTINE CALLS

Normally Sort functions are requested in the following order:

1. Initialize sort
2. Release record
3. Commence sort
4. Return record

The abort sort function may be requested at any point between sort initialization and the return of the last record.

The release record call must be issued for each record to be sorted. The return record call must be executed for each sorted record to be returned.

Only certain sequences of calls are permitted:

Previous Call	Calls That May Follow
None	Initialize Sort
Initialize sort	Release record, commence sort, abort sort
Release record	Release record, commence sort, abort sort
Commence sort	Return record, abort sort
Return record	Return record, abort sort
Abort sort or return record	Initialize Sort

PROGRAM PREPARATION

LINKING REQUIREMENTS FOR MOD 400

The application program that contains the Sort subroutine calls must be explicitly linked to the interface module ZS_INT.O.

Example 1:

In this example, the Linker is loaded, and the application module APP is linked to the interface module ZS_INT to form the bound unit BOUND.

```
LINKER BOUND
LINK APP,ZS_INT
MAP
QUIT
```

Example 2:

This example illustrates how to compile a COBOL application module CCB201 and link it to the interface module ZS_INT. In this example there is a request to change to the working directory identified by the pathname ^ SRTVL5>SETC210. The source module CCB201 is located within the working directory. The COBOL command includes the list data map option. The LINKER command designates that a bound unit named CCB201 will be created, LAF mode will be in effect, and Linker output will be produced on the printer. The directory ZCRT contains COBOL runtime routines. The interface module ZS_INT.O is located in SYSLIB2. In the following interactive dialog, entries supplied by the user are labeled (u) and those supplied by the system are labeled (s) in the margin to the left of the listing. (These labels do not appear on an actual listing.)

```
(u) CWD ^SRTVL5>SETC210
(s) RDY:
(u) COBOL CCB201  -LD  -COUT>SPD>LPT00
(s) COBOL 0201
(s) 0000 ERRORS
(s) END COMPILATION
(s) RDY:
(u) LINKER CCB201  -COUT>SPD>LPT00  -LAF
```

```
(s) LINKER 210  BU=CCB201   LINKED ON: 1977/11/02  1846:42.1  -LAF
(u) LIB ^ZSYS51>SYSLIB2>ZCRT;LIB2 ^ZSYS51>SYSLIB2;
(u) LINK CCB201,  ZS_INT;MAP;QT
(s) LAF OR SLIC ZSSRT.O  NT FND   )
(s) LAF OR SLIC ZSREL.O  NT FND   |
(s) LAF OR SLIC ZSCOMM.O  NT FND  } THESE ERROR MESSAGES CAN BE IGNORED
(s) LAF OR SLIC ZSRET.O  NT FND   |
(s) LAF OR SLIC ZSEND.O  NT FND   )
(s) ROOT  CCB201
(s) LINK DONE
(s) RDY:
```

Note:

The error messages occur because the Sort Subroutines are secondary entry points with the volume ZS_INT. The compiler may have generated embedded link references to the subroutine or object units. This does not result in unresolved SYNREFs.

LINKING REQUIREMENTS FOR MOD 600

The application program that contains the Sort subroutine calls must be explicitly linked to the interface module ZS_INM.O.

Example:

In this example, the Linker is loaded and the application module APP is linked to the interface module ZS_INM.O to form the bound unit BOUND.

```
LINKER
NAME BOUND
LINKS    APP,ZS_INM
MAP
QUIT
```

Since the Sort subroutines are secondary entry points within the module ZS_INM, the compiler may have generated embedded links referencing the subroutines as object units. This does not result in unresolved SYMREFs, and the associated warning message can be ignored. The format of the warning message is:

```
LINKER: (110209) NAMED FILE OR DIRECTORY NOT FOUND
****** UNABLE TO GET CU FILE
```

Associated with these warning messages, the link map will have embedded link entries referring to the following "object units": ZSSRT, ZSREL, ZSRELD, ZSCOMM, ZSRET, ZSRETD, and ZSEND.

FILE REQUIREMENTS

Be sure that SORTC, the Sort bound unit, resides in a library that can be located via the system's search rules (see the LIST SEARCH RULES command (LSR) in the *Commands* manual).

Sort requires a temporary or permanent disk work file. The work file size must be approximately 1.2 times the size of the average record multiplied by the number of records to be sorted.

If a temporary work file is used, it is dynamically created on the volume containing the current working directory. It is released at the completion of the sort.

If the work file is permanent, the work file LFN must be associated with the work file pathname before the sort is initiated. When the sort is completed, the work file is removed by Sort via the REMOVE-FILE macro call, which is described in the *System Service Macro Calls* manual.

During execution, Sort issues error messages to the error-out file.

RETURN CODES AND ERROR MESSAGES

Return codes are issued as 16-bit binary integers to report status and errors. Return codes and their meanings are given below. Unless noted otherwise, an appropriate message also is issued to the error-out file. As indicated below, most return codes are associated with particular sort functions. Within a sort function, a return code may have more than one meaning.

Usage	Decimal Representation of Return Code	Meaning
General	0	Requested function has been performed successfully (no message is issued).
	1	System error.
	2	Error in call sequence.
Initialize sort	1	Work file not found. Open error on work file. Value or length inconsistency — rightmost key byte. Linkage error; Sort cannot be executed.
	2	Invalid argument.
Release record	1	Work file too small. Read error on work file. Write error on work file.
	3	Record too small (no message is issued).
	4	Record too large (no message is issued).
Commence sort	1	Read error on work file. Write error on work file. Linkage error; Sort cannot be executed.
Return record	1	Read error on work file. Write error on work file. Data gain sort error. Output records from sort are out of sequence. Data has been lost within sort. Close error on work file.
	5	Sort has been terminated. The last record has been returned by the current return call (no message is issued).
	6	The last record has been returned, but the sort has not been successfully terminated; e.g., close error on work file or failure to return the memory used by Sort.
Abort sort	1	Linkage error; Sort cannot be executed. Close error on work file.

EXAMPLE OF SORTING USING SUBROUTINE CALLS

This example illustrates a COBOL source program that includes Sort subroutine calls.

```
GCOS6    COBOL    VERSION S200                    01/01/01    0000    PAGE 0001
 SOURCE  PROGRAM

    1          IDENTIFICATION DIVISION.
    2          PROGRAM-ID. SORTEX.
    3          ENVIRONMENT DIVISION.
    4          CONFIGURATION SECTION.
```

```
 5          SOURCE-COMPUTER. LEVEL-6.
 6          OBJECT-COMPUTER. LEVEL-6.
 7          INPUT-OUTPUT SECTION.
 8          FILE-CONTROL.
 9              SELECT INPUT-FILE
10              ASSIGN TO 0D-MSD
11              ORGANIZATION IS SEQUENTIAL
12              WITH VLR
13              ACCESS MODE IS SEQUENTIAL
14              FILE STATUS IS FILE-STATUS.
15              SELECT OUTPUT-FILE
16              ASSIGN TO 0E-MSD
17              ORGANIZATION IS SEQUENTIAL
18              WITH VLR
19              ACCESS MODE IS SEQUENTIAL
20              FILE STATUS IS FILE-STATUS.
21          DATA DIVISION.
22          FILE SECTION.
23          FD  INPUT-FILE
24              LABEL RECORD IS STANDARD.
25          01  INPUT-RECORD.
26           02 1ST-BYTE PIC X.
27           02 SIZE-BYTES PIC 99.
28           02 FILLER PICTURE X(77).
29          FD  OUTPUT-FILE
30              LABEL RECORD IS STANDARD.
31          01  OUTPUT-RECORD.
32           02 FILLER PIC X(80).
33          01  60BYTE-RECORD.
34           02 FILLER PIC X(60).
35          01  80BYTE-RECORD.
36           02 FILLER PIC X(80).
37          WORKING-STORAGE SECTION.
38          01  FILE-STATUS PICTURE XX VALUE SPACE.
39          01  SORT-SPACE.
40           02 FILLER COMP-1 VALUE +4096.
41           02 FILLER COMP-1 OCCURS 20 TIMES.
42          01  SORT-PARAMETERS.
43           02 MEMORY-SIZE PIC 99 VALUE 10.
44           02 FILL PIC X VALUE "N".
45           02 WORK-FILE-LEN PIC XX VALUE "00".
46           02 RECORD-TYPE PIC X VALUE "V".
47           02 MAX-RECORD-SIZE PIC 99999 VALUE 80.
48           02 CHARACTER-TYPE PIC XX VALUE "CH".
49           02 KEY-SIZE PIC 999 VALUE 5.
50           02 KEY-BEGINS PIC 99999 VALUE 4.
51           02 SORT-ORDER PIC X VALUE "A".
52           02 DESC-END PIC XXX VALUE "   ".
53          01  SORT-CALL-NAME REDEFINES SORT-PARAMETERS
54              PIC X(15) .
55          01  RECORD-LENGTH COMP-1 VALUE ZERO.
56          01  RETURN-CODE USAGE COMP-1.
57          PROCEDURE DIVISION.
58          FIRST-PARA.
59              OPEN INPUT INPUT-FILE.
60              CALL "7SSRT" USING SORT-SPACE RETURN-CODE SORT-CALL-NAME.
61          READ-PARAGRAPH.
62              READ INPUT-FILE AT END GO TO SORT-PARAGRAPH.
63              MOVE SIZE-BYTES TO RECORD-LENGTH.
64              IF 1ST-BYTE EQUAL "A"  GO TO READ-PARAGRAPH.
65              CALL "7SREL" USING SORT-SPACE RETURN-CODE
66                  INPUT-RECORD RECORD-LENGTH.
67              IF RETURN-CODE NOT EQUAL ZERO GO TO END-SORT-PARAGRAPH.
68              GO TO READ-PARAGRAPH.
69          SORT-PARAGRAPH.
70              CALL "7SCOMM" USING SORT-SPACE RETURN-CODE.
71              IF RETURN-CODE NOT EQUAL ZERO GO TO END-SORT-PARAGRAPH.
72          OUTPUT-PARAGRAPH.
73              OPEN OUTPUT OUTPUT-FILE.
74          WRITE-PARAGRAPH.
75              CALL "7SRET" USING SORT-SPACE RETURN-CODE
76                  OUTPUT-RECORD RECORD-LENGTH.
77              IF RETURN-CODE EQUAL TO 1 GO TO END-SORT-PARAGRAPH.
78              IF RETURN-CODE EQUAL TO 2 GO TO END-SORT-PARAGRAPH.
```

```
    79                IF RECORD-LENGTH EQUAL 60 WRITE 60BYTE-RECORD.
    80                IF RECORD-LENGTH EQUAL 80 WRITE 80BYTE-RECORD.
    81                IF RETURN-CODE  EQUAL TO 5 GO  TO LAST-PARA.
    82                GO TO WRITE-PARAGRAPH.
    83            END-SORT-PARAGRAPH.
    84                DISPLAY "ABNORMAL END-OF-SORT".
    85                DISPLAY "RETURN-CODE IS -> " RETURN-CODE.
    86            LAST-PARA.
    87                CLOSE INPUT-FILE.
    88                CLOSE OUTPUT-FILE.
    89                STOP RUN.
    90            END COBOL.
NO DIAGNOSTICS
```

Appendix A

Sort and Merge Memory Requirements

SORT MEMORY USAGE FOR MOD 200 AND MOD 400

Sort has the following memory requirements:

- Executable code: 4K words.
- Input buffer requirements: i words. For disk files to be used exclusively by Sort, i equals the control interval size; if the files are shareable, i equals 0. For tape, i equals the amount of data between interblock gaps.
- Task group overhead: 1K words.
- Work area of w words.

The total Sort memory requirement(s) is calculated by the following:

$$s = 5K + w + i$$

MINIMUM SORT MEMORY REQUIREMENTS FOR MOD 200 AND MOD 400

The minimum requirement for w is given by the formula:

$$w = \frac{5(r + 14)}{2} \text{ words}$$

where r represents the maximum input record size (in bytes). If the record is variable in length, four bytes should be added to the value of r. Sort performance is enhanced by allowing w to be larger than the minimum.

When the ARRange statement is used, the minimum requirement for w is given by the formula:

$$w = \frac{5(a+14) + r}{2}$$

where a is the size, in bytes, of the arranged record, and r is the maximum input record size, in bytes.

The control argument -SIZE n (or -SZ n) in the SORT command specifies the maximum amount of memory to be allocated to the Sort to support the Sort executable code and the work area w, as follows:

$$n = 4K + w$$

If the value specified in the -SIZE argument results in a requirement for more memory than the amount of memory available in the memory pool associated with the task group, Sort will not be aborted provided sufficient memory is available for the Sort to be executed.

The recommended minimum value for -SIZE is 8. You may, if necessary, attempt to specify smaller values of memory size down to a lower minimum of 6; in this case, however, using the formula given under "Sort Work File" (see above) to calculate work file size may result in underestimation of the space required.

INPUT/OUTPUT FILE BUFFERS FOR MOD 200 AND MOD 400

Unless modified by the GET command, the files are treated as nonshareable. Each file requires the full amount of memory from the task group memory pool: control interval size in words plus thirty-eight words. Since the input file is released before the output file is opened, the memory requirement for both input and output files does not occur simultaneously.

MERGE MEMORY USAGE FOR MOD 200 AND MOD 400

Merge has the following user memory requirements:

- Executable code: 5K words.
- Input file support: If the input files are not shareable, each input file requires the control interval size of the file (in words) plus 52 words; otherwise, no user memory is required.
- Output file support: If the output file is not sharable, it requires the control interval size (in words) plus 52 words of memory; otherwise, no user memory is required.
- Work area: The amount of memory required depends on whether there is an ARRange statement. In the following formulas,

  w = Work area
  n = Number of files to be merged
  i = Size of largest record to be merged
  r = Size of output record
  a = Size of the "arranged" record

  - No ARRange statement:
    - Fixed-length records, w = n(i)+r
    - Variable-length records, w = n(i+1) + (r+1)
  - ARRange statement included:

    w + n(a)+i+r
- Task overhead: 1K words.

SORT SUBROUTINE MEMORY USAGE FOR MOD 400

Sort requires a minimum of 8K words of memory more than that required by the application program. In the Sort Description of the initialize sort call, the size field specifies the amount of memory the sort will attempt to use: 4K is used by Sort subroutines, and the remainder is dynamically requested as a work area. The amount of memory actually obtained from the user's pool must be sufficient to support the sort. The minimum size of the work area (w) is determined by the following formula:

$$w = \frac{5(r+14)}{2} \text{ words}$$

r = Maximum input record size, in bytes.

Note:

If the record is variable in length, add 4 bytes to the value r.

If the work area is larger than the minimum required size, Sort performance is enhanced.

When determining a value for the size field, consider any increase in user memory requirements due to the actions of the application program after the sort has been initialized. For example, if you open a nonsharable output file between sort initialization and termination, the size value must be restricted so there is sufficient user memory space for necessary data management buffer and control structures.

SORT SUBROUTINE LRN'S (MOD 400 ONLY)

The user task group must allow for an LRN entry for the Sort.

SORT MEMORY USAGE FOR MOD 600

The maximum amount of task private memory to be used by Sort is specified in the -SIZE argument of the SORT command. Task private memory is used for the Sort executable code (bound unit) and the Sort work area:

- Executable code: 4K words.
- Work area: Space used by Sort for record storage.

  Note:

  This work area is not taken from the group work area.

The recommended minimum -SIZE value is 8K, but a sort with limited functionality that uses small records (less than 100 bytes) can be successfully run in 6K words of memory. However, if less than the minimum recommended value is specified, the formula for the disk work file space (see below) probably will underestimate the space required. There are two reasons for using a size greater than 8K: (1) Sort's performance will be enhanced and (2) records to be sorted may be too large for Sort to be executed. If the value specified in the -SIZE argument is larger than the task private memory, Sort will not be aborted provided sufficient memory is available for Sort to be executed.

MINIMUM SORT MEMORY REQUIREMENTS FOR MOD 600

In the following formulas:

s = Minimum value for -SIZE argument of SORT command.
w = Sort work area factor.
r = Sort record size, in bytes.
i = Input record size, in bytes.
a = Arranged record size, in bytes.

Calculating minimum value of -SIZE argument:

$s = 4096 + 256(w)$

The size of Sort's work area factor (w) depends on whether there is an ARRange statement.

- No ARRange statement:

$$w = \frac{4 + 5\,(r + 14)}{512} \text{ words rounded up}$$

  — UFAS files:
  If i is even, r = i + 4
  If i is odd, r = i + 5
  — Non-UFAS files (fixed relative):
  If i is even, r = i
  If i is odd, r = i + 1

- ARRange statement included:

$$w = \frac{4 + 5\,(r + 14) + i}{512} \text{ words rounded up}$$

  If a is even, r = a
  If a is odd, r = a + 1

Example:

The input file to be sorted has an input record size of 2003 bytes and the file organization is UFAS. There is no ARRange statement.

$$w = \frac{4 + 5\,(2008 + 14)}{512} = 24$$

$s = 4096 + 256(24) = 10{,}240$

The minimum value for the -SIZE argument of the SORT command is 10.

INPUT AND OUTPUT FILE BUFFERS FOR MOD 600

Unless modified by the GET command, the files are treated as nonsharable. Each file requires the following amount of memory in the group system area: the control interval size (in words) plus 38 words. Since the input file is released before the output file is opened, the memory requirements for both the input and output files do not occur simultaneously.

MERGE MEMORY USAGE FOR MOD 600

The major difference between Sort and Merge memory requirements is that Merge requires a specific amount of memory; therefore, there is no -SIZE argument in the MERGE command. Also, Merge does use the group work area as its work area.

Merge has the following memory requirements:

- Executable code: 4K words (task private memory).
- Input files: Unless modified by a GET command, the input files are nonsharable. Each input file requires in the group system area the control interval size (in words) plus 38 words.
- Output file: The output file usually is exclusive; therfore, memory must be available within the group system area for the output file's control interval size plus 38 words.
- Work area: The work area for Merge is taken from the group work area.

The size of the work area is determined by the following formulas. In these formulas,

n = Number of files to be merged.
i = Size, in bytes, of largest record to be merged.
r = Size, in bytes, of output record.
a = Size of arranged record.
w = Work area factor.

Note:

If i, r, or a is odd, add 1 to that value.

Calculating size of Merge's work area:

work area = 256(w)

The size of the work area factor (w) depends on whether there is an ARRange statement.

- No ARRange statement:
  — UFAS input files:

$$w = \frac{4 + [n(i + 2) + r]}{512}$$

  — Non-UFAS input files:

$$w = \frac{4 + [n(i) + r]}{512}$$

- ARRange statement specified:

$$w = \frac{4 + [n(a) + i + r]}{512}$$

SORT SUBROUTINE MEMORY USAGE FOR MOD 600

The function of the size field in the sort description passed by the calling program is identical to that of the -SIZE argument of the SORT command. The size field specifies the maximum amount of task private memory to be used for the Sort executable code and the Sort work area. When using subroutine calls, Sort's usage of the task private memory is in addition to that required by the calling program. Since Sort is run as a separate task, a task stack area of 1024 words is required.

If the value specified for the size field exceeds what is available in the task private memory, Sort execution will continue if the amount of memory obtained is sufficient for Sort to be successfully executed. The minimum value for the size field is 8K.

In the following formulas:

s = Minimum value of size field.
w = Sort work area factor.
b = Bound unit (4K words).
r = Sort record size.
i = Input record size (record released to Sort).

The minimum size value is determined by the following formula:

$$s = b + 256(w)$$

The minimum requirement for w is:

$$w = \frac{4 + 5\,(r + 14)}{512} \text{ words rounded up}$$

The value of the Sort record size (r) depends on the type of records.

- Fixed-length records:
  If i is even, r = i
  If i is odd, r = i + 1
- Variable-length records:
  If i is even, r = i + 4
  If i is odd, r = i + 5

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 6) GCOS 6 SORT/MERGE ADDENDUM B

ORDER NO. CB04B, REV. 0

DATED NOVEMBER 1978

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

CUT ALONG LINE

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

FOLD ALONG LINE

Honeywell