SPERRY RAND

UNIVAC 9400 SYSTEM

SORT/MERGE

PROGRAMMERS REFERENCE

A

UP-7664

This manual is published by the Univac Division of Sperry Rand Corporation in loose leaf format. This format provides a rapid and complete means of keeping recipients apprised of UNIVAC® Systems developments. The information presented herein may not reflect the current status of the product. For the current status of the product, contact your local Univac Representative.

The Univac Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of hardware or software changes and refinements. The Univac Division reserves the right to make such additions, corrections, and/or deletions as, in the judgment of the Univac Division, are required by the development of its Systems.

PRINTED IN U.S.A.

UNIVAC 9400 System
SORT/MERGE Programmers Reference
UP-7664

April 27, 1970

UPDATING PACKAGE "A"

UNIVAC 9400 System P.I.E. Bulletin 12, UP-7593.12, announces the release and availability of Updating Package "A" to "UNIVAC 9400 System SORT/MERGE Programmers Reference," UP-7664, page i and 47 pages plus 1 Updating Summary Sheet.

SECTION	DESTROY FORMER PAGES NUMBERED	FILE NEW PAGES NUMBERED
Front Cover & Disclaimer	†	†
Contents	1 and 2	1 Rev. 1 and 2 Rev. 1
	N. A.	3**
Section 1	1 and 2	1 Rev. 1 and 2*
Section 2	1 and 2	1* and 2 Rev. 1
	5 and 6	5 Rev. 1 and 6*
	9 and 10	9 Rev. 1 and 10 Rev. 1
	11 and 12	11* and 12 Rev. 1
	13 and 14	13 Rev. 1 and 14*
	15 and 16	15* and 16 Rev. 1
Section 3	1 and 2	1 Rev. 1 and 2 Rev. 1
Section 4	1 and 2	1 Rev. 1 and 2 Rev. 1
Section 5	3 and 4	3 Rev. 1 and 4 Rev. 1
Section 6	3 and 4	3 Rev. 1 and 4 Rev. 1
Section 7	1	1 Rev. 1
Appendix A	N. A.	1** thru 5**
Appendix B	N. A.	1** thru 3**
Appendix C	N. A.	1** and 2**
Appendix D	N. A.	1**
Appendix E	N. A.	1** thru 4**
Appendix F	N. A.	1** thru 4**

†Destroy old cover and file new cover.
*These are backups of revised pages and remain unchanged.
**These are new pages.

CONTENTS

1. INTRODUCTION

1.1. GENERAL

This manual describes the Sort/Merge facilities provided for the UNIVAC 9400 System. It includes descriptions of the acceptable sort conventions and the linkage commands required to perform a sort. Use of this manual assumes knowledge of *UNIVAC 9400 Job Control for Disc Systems Programmers Reference, UP-7585* (current version), *UNIVAC 9400 Assembler/CPU Programmers Reference, UP-7600* (current version), and *UNIVAC 9400 Supervisor Programmers Reference, UP-7689* (current version).

The UNIVAC 9400 Sort/Merge program is a modular type of subroutine integrated into a problem program. The subroutine concept provides a highly efficient and comprehensive sorting capability that can be applied to a wide range of data processing requirements. Specifications of the sort are achieved through parameters which indicate the particular needs of a specific sort run. Significant features of the UNIVAC 9400 Sort/Merge program are:

- A capability that allows an unlimited volume of data to be sorted, either automatically or in separate phases.
- The ability to sort either fixed-length or variable-length records.
- The handling of five types of key field formats.
- The sorting of key fields in ascending or descending sequence.
- The permitting of execution of record sequence own code.
- The permitting of execution of data reduction own code.
- The providing of rerun facilities.

Merging is an integral function of sorting. Therefore any references to the Sort subroutine includes the merge function.

Before the Sort subroutine can be executed, the problem program must generate a table containing the parameters defining the sort run. When a linkage command is issued by the problem program, the proper module of the Sort subroutine is activated, and it has available to it such parameter information as names, addresses, number of bytes, and the type of sort run. When the operation of the particular module is completed, the Sort subroutine returns control to the problem program. The Sort subroutine communicates with the problem program by means of the general registers. Messages that require operator action are routed to the console printer.

1.2. FUNCTIONAL DESCRIPTION

The Sort subroutine is capable of sorting and merging small or large volumes of data. The sorting procedure is functionally described as a sequence of operational phases. Figure 1–1 is a block diagram of the operational phases for both small and large volume sorts.

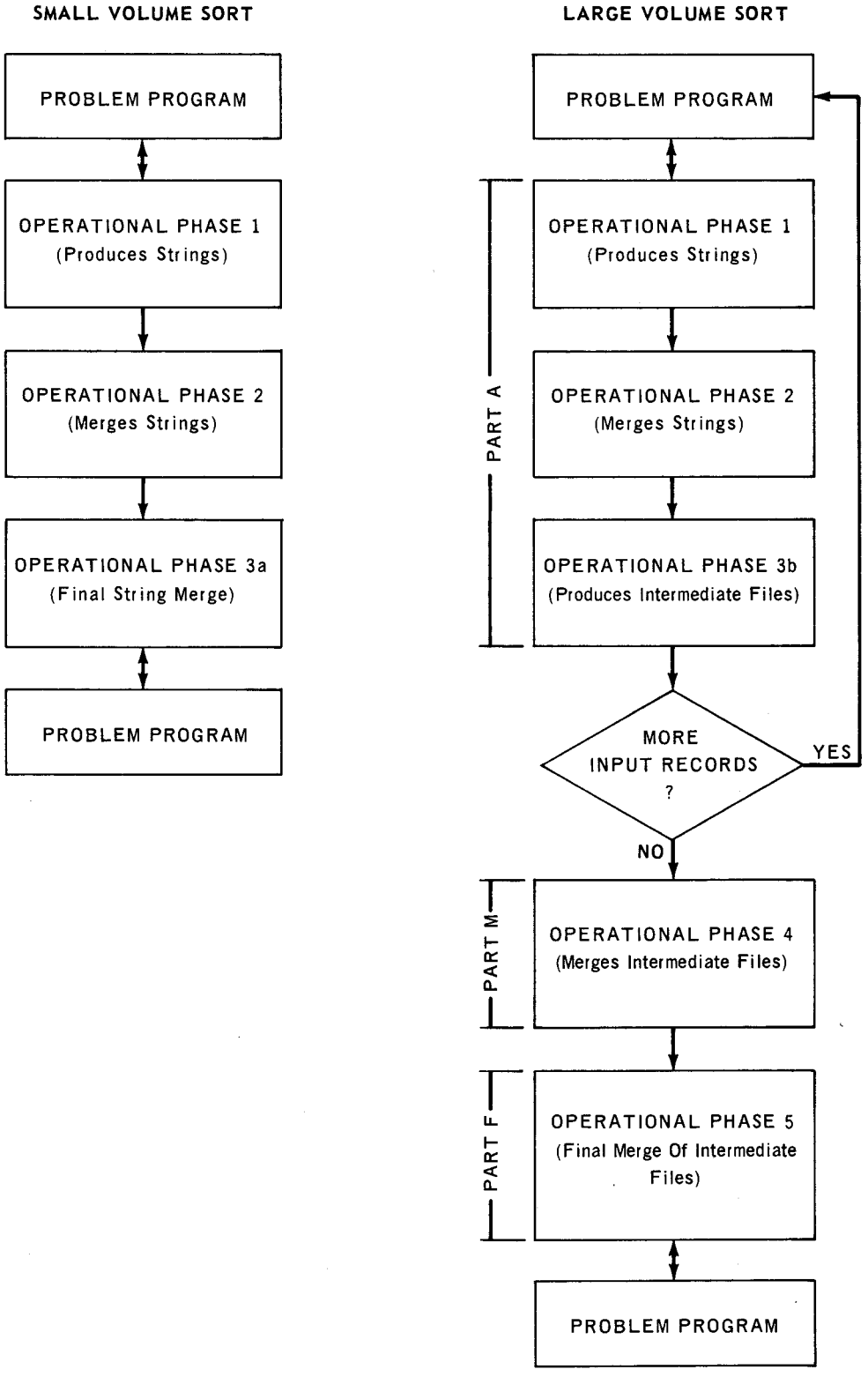

Figure 1–1. Block Diagram of Small and Large Volume Sort Operational Phases

1.2.1. Small Volume Sorts

A *small volume sort* requires no operator intervention and is performed in a single execution of operational phases 1, 2, and 3a.

During operational phase 1 of a small volume sort, the Sort subroutine accepts successive records, one at a time, from the problem program. If internal storage is exceeded, the strings are stored on disc or magnetic tape. The acceptance of records continues until the problem program indicates that all records are released to the Sort subroutine.

Operational phase 2 of the Sort subroutine collates the strings of sequenced records into longer strings. This collation is repeated until a final collation would produce one sorted file.

The final function (operational phase 3a) of the Sort subroutine performs a final merge of the strings. As the final merge is performed, the sorted records are returned to the problem program, one record at a time.

1.2.2. Large Volume Sorts

A *large volume sort* requires operator intervention to load, unload, and label tapes, and is performed in the execution of the operational phases 1, 2, 3b, 4, and 5.

During operational phase 1 of a large volume sort, the Sort subroutine accepts successive records, one at a time, from the problem program. If internal storage is exceeded, the strings are stored on disc or magnetic tape. Records are accepted until the magnetic tape capacity of the sort cycle is reached or the problem program requests a cycle break.

Operational phase 2 of the Sort subroutine collates the strings of sequenced records into longer strings. This collation is repeated until a final collation would produce one sorted string.

Operational phase 3b of the Sort subroutine performs the final merge of strings and produces an intermediate sorted file stored on magnetic tape. Such intermediate sorted files are produced until all records are released to the Sort subroutine.

Operational phase 4 merges the intermediate sorted files into longer intermediate files. This merging is repeated until a final intermediate merge would result in one sorted file.

Operational phase 5 performs the final merge of intermediate files. As the final merge is performed, the sorted records are returned to the problem program, one record at a time.

A large volume sort has three distinct parts which may be executed as an entity or as individual and separate parts. The two forms of execution are termed automatic or nonautomatic, respectively. The first part, Part A, produces intermediate sorted files. The second part, Part M, merges the intermediate files produced in Part A into larger intermediate files. The third part, Part F, merges the intermediate files from Part M and returns the records as one final sorted file to the problem program.

1.3. TAPE/DISC REQUIREMENTS

The Sort subroutine interface permits operation of disc-only, tape-only, or tape-disc sorts.

For a disc-only sort, the disc storage available to the sort must be large enough to contain the entire file of records to be sorted plus sort control information.

For a tape-only sort, a minimum of three tape units is required. A maximum of 14 tape units may be assigned. Operational phases 1, 2, and 3a or 3b utilize up to six tape units. Operational phases 4 and 5 for a large volume sort may use up to 14 tape units.

For a tape-disc sort, tape requirements are the same as stated for a tape-only sort. Disc storage is used to increase the length of the strings before tape collation. Disc storage reserved for the Sort subroutine must be large enough to contain a minimum of two strings produced by operational phase 1.

1.4. STATEMENT CONVENTIONS

The conventions used to illustrate statements in this manual are as follows:

- Capital letters and punctuation marks (except braces, brackets, and ellipses) indicate information that must be coded exactly as shown.
- Lowercase letters and terms represent information that must be supplied by the user.
- Information contained within braces { } represents necessary entries, one of which must be chosen.
- Information contained within brackets [ ] represents optional entries that are included or omitted (depending on program requirements).

  Braces within brackets signify that one of the entries must be chosen if that operand is included.
- Ellipses (series of three periods) indicate the presence of a variable number of entries.
- The operation codes for the Sort subroutine linkage macro instructions have the form

  MR$xxx

  where xxx represents the function associated with the particular macro instruction.
- The operand field of the macro instruction used to generate the Sort parameter table requires the following additional conventions:

  – Keyword parameters may appear in any order and must be separated by commas.

  – When a keyword parameter has a series of specifications, the series must be enclosed in parentheses, and the specifications must be separated by commas.

  – If a specification is omitted within a series, the comma must be retained to indicate the omission, except in the case of trailing commas.

1.5. SORT SUBROUTINE GENERAL REGISTER USAGE

The problem program and Sort subroutine communicate through general registers. For Sort subroutine linkage coding (see 4.1) general registers 0 and 1 are used for passing parameter information. Registers 14 and 15 are used as linkage registers.

Registers 11, 12, 14, and 15 are used for communication between the Sort subroutine and user own code data reduction or comparison routines (see 3.1). Registers 11 and 12 are used to transmit record addresses.

2. SORT FILE DEFINITION

2.1. GENERAL

The problem program must define the particular sort run for the Sort subroutine. This is accomplished through a Sort parameter table. When the problem program specifies the MR$PRM macro instruction a Sort parameter table is generated, or the problem program may construct the Sort parameter table internally. Either method may be used to partially construct the table and the table can be completed by submitting parameters to the Sort subroutine at object time by means of the control stream.

Keyword parameters associated with the MR$PRM macro instruction become entries in the Sort parameter table. These entries may also be set either by the problem program or through the control stream. Therefore, any reference in this manual to a keyword parameter also refers to a Sort parameter table entry.

2.2. MR$PRM MACRO INSTRUCTION

The Sort subroutine is activated at object time by executing the Sort initialization linkage MR$OPN (see 4.3). When the initialization linkage is executed, the problem program must specify a set of parameters which define the particular sort run. The MR$PRM macro instruction may be used to generate the Sort parameter table from the set of specified parameters. The table must be located on a word boundary and consists of a variable number of contiguous entries, one entry for each parameter.

The format of the MR$PRM macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[param-table-name]	MR$PRM	keyword-param-1[,keyword-param-2,..., keyword-param-n]

The label is optional and may consist of a maximum of eight alphanumeric characters. The keyword parameters may appear in any order but must follow the conventions stated in 1.4.

2.3. KEYWORD PARAMETERS

The keyword parameters which may be used to define a particular sort run are described in the following paragraphs. A summary listing of the keyword parameters which may appear in the operand of the MR$PRM macro instruction is given in Table 2–1.

KEYWORD	SPECIFICATIONS
RCSZ	maximum-bytes
BIN	{bytes (min-bytes,size-1,vol-1,...,size-n,vol-n)}
FIELD	(byte-pos-1,lgth-1[,form-1][,seq-1][,order-1],...,byte-pos-n, lgth-n[,form-n][,seq-n][,order-n])
RSOC	symbol
DROC	{symbol DELETE}
STOR	{symbol (symbol,number-of-bytes)}
IN	symbol
OUT	symbol
FIN	symbol
TAPES	{label-type (label-type,max-filenumber)}
SHARE	sort-filename
RESERV	sort-filename
DISC	max-disc-filenumber
PAD	bytes
ADTABL	symbol
AUTO	(label-code,sort-filename[,rec-per-cycle])
DOEXT	symbol
DOA	(label-code,sort-filename[,rec-per-cycle][,number-merge-tape])
DOM	(label-code,sort-filename,cycle-number,in-label-1, in-label-2[,...,in-label-13])
DOF	(in-label-1[,in-label-2,...,in-label-14])
RESUME	{(PASS,recovery-number) (CYCLE,cycle-number, from-rec-number) (MERGE,tape-label,reel-number) (FINAL,symbol)}
REDO	{(CYCLE,cycle-number,from-rec-number,to-rec-number) (MERGE,tape-label,reel-number,to-rec-number)}
NOCKSM	{device (device,device)}
CSPRAM	{YES NO OPTION}

Table 2–1. Summary of Keyword Parameters for the MR$PRM Macro Instruction

2.3.1. Record Size

The keyword parameter defining the record size is required. It describes the size of the fixed-length records or the maximum size of variable-length records of the data to be sorted. If the record format is variable length, the number of bytes must include the four-byte record length field that precedes each record. The maximum record size may not exceed 16,383 bytes.

The format for the keyword parameter is:

RCSZ=maximum-bytes

where:

maximum-bytes is the decimal number specifying the number of bytes for fixed-length records or the maximum number of bytes for variable-length records.

2.3.2. Bin Size for Variable-Length Records

The BIN keyword parameter is required for variable-length records. In order to conserve main storage and provide optimum speed in operational phase 1, variable-length records are divided into fixed bin sizes (fixed-length subrecords). The first bin must contain all key fields within the record. The BIN keyword parameter may be specified in either of two forms.

- Number of Bytes

  Bin size may be specified as the number of bytes into which a variable-length record can be divided. The format is:

  BIN=bytes

  where:

  bytes is the decimal number specifying the number of bytes of the bin (fixed-length subrecord) into which a variable-length record is to be divided.

- Distribution of Record Lengths

  The size of the bin may be determined by the distribution of the record lengths. The Sort subroutine selects the bin size in accordance with the record length and frequency of each length within the total number of records to be sorted. The format is:

  BIN=(min-bytes,size-1,vol-1,...,size-n,vol-n)

  where:

  min-bytes — is the decimal number specifying the minimum number of bytes of the bin into which a variable-length record can be divided. The number of bytes specified must accommodate all key fields of the record.

  size-i — is the decimal number specifying the record length in bytes.

  vol-i — is the decimal number specifying the frequency of size-i records within the total number of records to be sorted. The range of values for vol-i is from 0 to 100.

  For each variation in size (size-i) there must be a corresponding frequency (relative volume, vol-i) of the total number of records to be sorted.

2.3.3. Key Field Description

The FIELD keyword parameter is required if the Sort subroutine is to perform all key field record comparisons. The byte positions of a record are numbered, starting with 0, from low order to high order main storage (most significant to least significant byte). The major key field of a record is numbered 1, the next most significant key field is 2, and so on. The maximum number of key fields that can be specified is 255.

For variable-length records, the four-byte record length field is considered to be part of the record. All key fields of the record must be contained within the bin specified by the BIN keyword parameter.

If one of the optional specifications is omitted, the comma must be retained to indicate the omission. Trailing commas are not required. The format of the FIELD keyword parameter is:

FIELD=(byte-pos-1,lgth-1[,form-1][,seq-1][,order-1],...,byte-pos-n,lgth-n
[,form-n][,seq-n][,order-n])

where:

byte-pos-i is a decimal number specifying the position of the most significant byte of the key field relative to the beginning of the record.

lgth-i is a decimal number specifying the length of the key field in bytes. The maximum length is dependent on the format of the key field.

form-i is a two-character alphabetic code indicating the format of the key field. The relation between the format codes and the maximum length is as follows:

Format Code	Maximum Length (in Bytes)
CH (character)	256
BI (binary)	256
FI (fixed point integer)	256
PD (packed decimal)	16
ZD (zoned decimal)	16

The form-i specification is optional; if omitted, the character format (CH) is assumed.

seq-i is an alphabetic character indicating the sorting sequence for the key field.

A = Ascending sequence
D = Descending sequence

The seq-i specification is optional; if omitted, ascending sequence (A) is assumed.

order-i is a decimal number specifying the significance of the record key fields from major to minor. The order-i specification is optional. If specified, order-i must be specified in all key descriptions. If omitted, it must be omitted in all key descriptions. Then the first key field parameter set is assumed to describe the major key; the next parameter set is assumed to describe the next most significant field, and so on.

2.3.4. Record Sequence Own Code Routine

The RSOC keyword parameter is specified when the problem program is to compare all key fields of all records. See 3.1.1 for a discussion of the record sequence own code routine.

The format for the RSOC keyword parameter is:

RSOC=symbol

where:

symbol — is the symbolic label of the entry address of the user routine to compare key fields.

2.3.5. Data Reduction Own Code Routine

When the problem program is to eliminate or combine records with equal key fields, the DROC keyword parameter is specified. See 3.1.2 for a description of the requirements of the data reduction own code routine.

- Problem Program Routine

  The format of the keyword parameter when equal records are to be processed by the problem program is:

  DROC=symbol

  where:

  symbol — is the symbolic label of the entry address of the user routine for data reduction.

- Eliminating Equal Records

  The format of the keyword parameter when one of two records with equal key fields is to be eliminated is:

  DROC=DELETE

  where:

  DELETE — directs the Sort subroutine to perform data reduction automatically.

2.3.6. Storage Allocation

The STOR keyword parameter is required. The number of bytes to be reserved in main storage for the Sort subroutine need not be specified; the starting address must be specified.

When the number of bytes is specified, the user must ensure that a contiguous area of storage of the size specified is reserved for the Sort subroutine. This area is used for both working storage and the Sort subroutine coding. If the number of bytes is not specified, the Sort subroutine uses main storage starting at the specified address up to the upper bound of storage allocated to the problem program. The format is:

$$\text{STOR}=\left\{\begin{matrix}\text{symbol}\\(\text{symbol,number-of-bytes})\end{matrix}\right\}$$

where:

symbol	is the symbolic label of the starting address of main storage available to the Sort subroutine.
number-of-bytes	is a decimal number indicating the maximum number of bytes of main storage reserved for the Sort subroutine beginning at the specified address. This specification is optional.

2.3.7. Initialization Return Address

The IN keyword parameter defines the address to which control is returned in the problem program after the Sort subroutine has been initialized following execution of the MR$OPN linkage (see 4.3). This keyword parameter is required and has the following format:

IN=symbol

where:

symbol	is the symbolic label of the starting address of user coding which is to be executed after the Sort subroutine has been initialized.

2.3.8. Sorting Completed Return Address

The OUT keyword parameter defines the address to which control is returned in the problem program after the Sort subroutine has completed sorting the records. Control is returned after MR$SRT linkage coding (see 4.5) has been executed and the Sort subroutine is ready to deliver the sorted data. The problem program may obtain the sorted records by executing the MR$RET linkage coding (see 4.6). The OUT keyword parameter is required and has the following format:

OUT=symbol

where:

symbol — is the symbolic label of the entry address in the problem program of coding to be executed when the Sort subroutine is ready to deliver records in the final sorted sequence.

2.3.9. End of Sort

The FIN keyword parameter is required and defines the address to which control is returned in the problem program after all sorted records have been delivered to the problem program. The records are delivered to the problem program by executing the MR$RET linkage coding. The keyword parameter has the following format:

FIN=symbol

where:

symbol — is the symbolic label of the entry address of user coding to be executed after all sorted records have been delivered to the problem program.

2.3.10. Scratch Tape Allocation

The TAPES keyword parameter specifies the number of magnetic tape units that may be used by the Sort subroutine as scratch tapes. The Sort subroutine requires a minimum of three scratch tapes. A maximum of 14 tape units may be assigned. A maximum of six tapes are used by the Sort subroutine for string collating (operational phase 1). For large volume sorts, up to 14 tapes may be used for intermediate file merging.

Physical tape units are assigned to the Sort subroutine by means of DVC Job Control statements. Sort tape files are assigned by standard sort filenames (SM01, SM02,..., SM14). These filenames are restricted for the exclusive use of the Sort subroutine and must be assigned in sequence beginning with SM01. For files with standard labels, DVC, VOL, and LFD Job Control statements should describe each file. For files with no labels, only the DVC and LFD Job Control statements are needed for each file. UNISERVO VI C, 12, or 16 Magnetic Tape Units may be assigned. Seven-level tapes must have the data conversion feature, and the mode must be assigned as 800 ppi, translator off, and data conversion on.

The TAPES keyword parameter has the following format:

$$\text{TAPES}=\left\{\begin{array}{l}\text{label-type}\\(\text{label-type,max-filenumber})\end{array}\right\}$$

where:

label-type indicates the label convention for all tape files assigned to the Sort subroutine. Enter either STD if Sort scratch tapes have standard labels, or NO if Sort scratch tapes are unlabeled.

max-filenumber is a decimal number indicating the maximum number of standard sort filenames which may be assigned (3 through 14). If not specified, a maximum of 14 sort filenames is assumed.

2.3.11. Shared Input Tape

The problem program may use a tape unit assigned to the Sort subroutine during operational phase 1 of a sort run. The SHARE keyword parameter may be specified only for a small volume sort. The tape unit assigned to the specified file is not used by the Sort subroutine until all records have been released to the Sort subroutine. The specified tape unit is then rewound with interlock and an operator request is issued to mount a scratch tape. The keyword parameter has the following format:

SHARE=sort-filename

where:

sort-filename is a standard sort filename.

2.3.12. Reserve Final Tape Unit

The RESERV keyword parameter is optional and may be specified only for a small volume sort. The tape unit assigned to the specified file is used by the Sort subroutine only during operational phases 1 and 2. The problem program may use the tape unit after control is returned to the address specified by the OUT keyword parameter. The tape is rewound without interlock by the Sort subroutine and an operator request is issued to mount the user reserve file tape. The format of the keyword parameter is:

RESERV=sort-filename

where:

sort-filename is a standard sort filename.

RESERV and SHARE parameters cannot specify the same sort filename.

2.3.13. Disc Allocation

The DISC keyword parameter indicates that disc units may be used by the Sort subroutine for scratch purposes. If only disc units are assigned, the disc area available to the Sort subroutine must be large enough to hold the entire file to be sorted plus sort control information. If both tape and disc units are assigned, the disc area must be large enough to hold at least two strings generated during operational phase 1 (approximately four times the storage area assigned to the sort). The format for this keyword parameter is:

DISC=max-disc-filenumber

where:

max-disc-filenumber	a decimal number (1 through 8) which indicates the maximum standard sort disc filenames which may be assigned to the Sort subroutine.

Physical disc units are assigned to the Sort subroutine by means of the DVC and LFD Job Control statements. Sort disc units are assigned by standard sort disc filenames (DM01, DM02,..., DM08). These filenames are restricted for the exclusive use of the Sort subroutine and must be assigned in sequence beginning with DM01. Up to four disc units may be assigned to each sort disc-filename. Each disc track is assumed to have a home address followed by the track descriptor record (R0).

2.3.14. Extension of Parameter Table

The PAD keyword parameter permits the user to augment the parameter table beyond the required length. Additional parameters can then be entered into the table by the problem program. This keyword parameter has the following format:

PAD=bytes

where:

bytes	is the number of bytes in decimal to be added to the parameter table.

2.3.15. Additional Parameter Table

The user may wish to specify an additional parameter table within the problem program or reference a previously created table. Either is done by means of the ADTABL keyword parameter. This parameter table may contain information that is similar to, but mutually exclusive of, the information contained in the parameter table generated by the MR$PRM macro instruction. If the ADTABL keyword parameter is specified, a sentinel branch entry is generated as the last item in the original parameter table. When the Sort subroutine is scanning table entries, a branch entry causes the Sort subroutine to branch to the referenced table and continue scanning entries in that table. If the referenced table is assembled separately, then the address specified must be defined as an external reference (EXTRN). The format of the keyword parameter is:

ADTABL=symbol

where:

symbol — specifies the symbolic label of the starting address of another Sort parameter table.

2.3.16. Automatic Execution of Large Volume Sort

The AUTO keyword parameter is used when the Sort subroutine is to execute all parts of a large volume sort automatically. (See Section 5 for additional information regarding large volume sorts.) The Sort subroutine controls the execution of the parts and provides instructions to the operator for labeling, loading, and unloading tapes. The AUTO keyword parameter has the following format:

AUTO=(label-code,sort-filename[,rec-per-cycle])

where:

label-code — is a two-character, alphanumeric user identification which is appended, with Sort identification, to form a label for each intermediate output file produced.

sort-filename — is a standard sort filename which designates the tape unit receiving all intermediate output cycle files.

rec-per-cycle — is a decimal number indicating the maximum number of records to be sorted in any cycle. This field is optional and allows the user to control the number of records to be sorted for each cycle of Part A.

2.3.17. Part A or Part M Return Address

The DOEXT keyword parameter is optional. It is applicable only for the nonautomatic execution of a large volume sort. At the completion of Part A or Part M of a large volume sort, the Sort subroutine returns control to the problem program at the specified address. The format of the keyword parameter is:

DOEXT=symbol

where:

symbol — is the symbolic label of the starting address of user coding to be executed after Part A or Part M of a large volume sort is completed.

2.3.18. Execute Part A

The DOA keyword parameter is used when Part A of a large volume, nonautomatic sort is to be executed. The number-merge-tape specification is optional; when specified, it is used by the Sort subroutine as a limiting factor in determining block sizes. The format of the keyword parameter is:

DOA=(label-code,sort-filename[,rec-per-cycle][,number-merge-tape])

where:

label-code — is a two-character, alphanumeric user identification which is appended, with Sort identification, to form a label for each intermediate output file produced.

sort-filename — is a standard sort filename which designates the tape unit receiving each intermediate output cycle file.

rec-per-cycle — is a decimal number indicating the maximum number of records to be sorted in any cycle. This field is optional and allows the user to control the number of records to be sorted for each cycle of Part A.

number-merge-tape — is a decimal number indicating the maximum number of standard sort filenames which may be assigned in merge runs. This specification is optional.

2.3.19. Execute Part M

The DOM keyword parameter is used when Part M of a large volume, nonautomatic sort is to be executed. The number of input files which can be merged is one less than the actual number of files assigned to the sort. The keyword parameter has the following format:

DOM=(label-code,sort-filename,cycle-number,in-label-1,in-label-2[,...,in-label-13])

where:

label-code — is a two-character, alphanumeric user identification which is appended, with Sort identification, to form a label for the intermediate output file produced.

sort-filename — is a standard sort filename which designates the tape unit receiving each intermediate output cycle file.

cycle-number — the output cycle number for the intermediate output file.

in-label-i — is the label of each intermediate input file to be merged into a single intermediate output file. Up to 13 input files may be specified. These labels are supplied to the user through a console message to the operator in a previous DOA or DOM run.

2.3.20. Execute Part F

The DOF keyword parameter is used when Part F of a large volume, nonautomatic sort is to be executed. Up to 14 input files may be specified. The format of the keyword parameter is:

DOF=(in-label-1[,in-label-2,...,in-label-14])

where:

in-label-i — is the label of each intermediate input file to be merged into a final output file. These labels are supplied to the user through console messages to the operator in previous DOA or DOM runs.

2.3.21. Resumption of an Interrupted Sort

The RESUME keyword parameter is used to continue executing an interrupted sort. Section 6 explains the use of the various forms of this keyword parameter. There are four possible points at which an interrupted sort may be resumed.

- Tape Collation

  A small or large volume sort may be resumed from the collation pass that was interrupted. The following keyword parameter must be added to the original set of parameters prepared for the sort.

  RESUME=(PASS,recovery-number)

  where:

  PASS — indicates resumption of a sort interrupted during tape collation.

  recovery-number — is the most recent collation pass recovery number supplied to the operator by means of a console message.

- Part A

A large volume sort interrupted during the execution of Part A may be resumed. The following keyword parameter must be added to the original set of parameters prepared for the sort.

RESUME=(CYCLE,cycle-number,from-rec-number)

where:

CYCLE	indicates resumption of a sort interrupted during Part A processing of a large volume sort.
cycle-number	is a decimal number to be assigned to the first cycle after restart.
from-rec-number	is the decimal number assigned to the first record of the first cycle after restart.

- Part M

A large volume sort interrupted during the execution of Part M may be resumed. The following keyword parameter must be added to the original set of parameters prepared for the sort.

RESUME=(MERGE,tape-label,reel-number)

where:

MERGE	indicates resumption of a sort interrupted during Part M processing of a large volume sort.
tape-label	is the label assigned to the first output tape to be written after restart.
reel-number	is the reel (volume) number assigned to the first tape written after restart.

- Part F

A large volume sort interrupted during the execution of Part F may be resumed. The following keyword parameter must be added to the original set of parameters prepared for the sort.

RESUME=(FINAL,symbol)

where:

FINAL	indicates resumption of a sort interrupted during the final merge (Part F) of a large volume sort.
symbol	specifies the symbolic label of the starting address of the rerun information table maintained by the Sort subroutine during the execution of Part F of a large volume sort.

2.3.22. Re-Creation of a Part A File or a Part M File

The REDO keyword parameter provides the capability of reproducing a Part A intermediate output file or a Part M output tape. The two uses of this keyword parameter are given in 6.6 and 6.7. The formats of this keyword parameter are:

- Part A File

  If an output reel produced during Part A processing cannot be read, the output file incorporating this reel can be re-created. The following keyword parameter must be added to the original set of parameters prepared for the sort.

  REDO=(CYCLE,cycle-number,from-rec-number,to-rec-number)

  where:

CYCLE	indicates that a Part A intermediate output file of a large volume sort is to be re-created.
cycle-number	is the decimal number assigned to the output file being re-created.
from-rec-number	is a decimal number specifying the number of the first record in the set of records being sorted again.
to-rec-number	is a decimal number specifying the number of the last record in the set of records being sorted again.

- Part M Tape

  An output reel produced during Part M processing of a large volume sort may be re-created. The following keyword parameter must be added to the original set of parameters prepared for the sort.

  REDO=(MERGE,tape-label,reel-number,to-rec-number)

  where:

MERGE	indicates that a Part M tape of a large volume sort is to be re-created.
tape-label	is the label of the Part M tape to be reproduced.
reel-number	is the reel (volume) number of the tape.
to-rec-number	specifies the last record number, in decimal, contained on the original tape.

2.3.23. Checksum Elimination

A checksum word is calculated and written for each output block that is written; the checksum word is verified for each input block that is read. The NOCKSM keyword parameter causes the checksum calculation to be bypassed for the indicated device. The format is:

$$\text{NOCKSM}=\left\{\begin{matrix}\text{device}\\(\text{device,device})\end{matrix}\right\}$$

where:

device — specifies the hardware device for which the checksum option is to be omitted.

T = omit tape checksum

D = omit disc checksum

2.3.24. Control Stream Parameter Option

Sort parameters may be entered through the system control stream (see 7.1). The CSPRAM parameter directs the Sort to access control stream parameters, bypass accessing control stream parameters, or allow the operator to indicate the presence of control stream parameters at run time. The format of this keyword parameter is:

$$\text{CSPRAM}=\left\{\begin{matrix}\text{YES}\\\text{NO}\\\text{OPTION}\end{matrix}\right\}$$

where:

YES — indicates Sort control stream parameters are to be accessed during Sort initialization.

NO — indicates Sort control stream parameter accessing should be bypassed.

OPTION — allows the operator to make the above YES or NO indication for control stream Sort parameters at run time. A console message requiring an operator response is issued at Sort initialization. If the CSPRAM parameter is not specified, OPTION is assumed by the Sort subroutine.

3. SPECIAL OWN CODE ROUTINES

3.1. GENERAL

The user may provide routines to compare all key fields of all records or to perform data reduction. The addresses to these routines may be supplied by the RSOC or DROC keyword parameters, respectively. Sort calls may not be executed from RSOC or DROC. The Sort subroutine transfers control to these addresses when specified. Both routines require registers 11, 12, 14, and 15 for communication. Only registers 11, 12, 14, and 15 contain valid information when entrance is made to own code routines. The contents of these registers must not be destroyed. All other registers are available for use within the own code. Boundaries need not be followed for records passed to or from the Sort.

3.1.1. Record Sequence Own Code Routine

The user can compare all key fields of all records with a record sequence own code routine. The address of the routine may be entered into the parameter table by means of the RSOC keyword parameter. (The FIELD keyword parameter should not be specified.) When it is necessary to decide which of two records is to precede the other, the Sort subroutine transfers control to the address specified by the RSOC keyword parameter. Registers 11 and 12 contain the addresses of the two records to be compared. Register 14 contains the Sort subroutine return address. Register 15 contains the address of the record sequence own code routine.

After the two records have been compared, the record sequence own code routine must set the Condition Code to indicate which record is to precede the other or whether they are equal. If the record addressed by register 11 is to precede, the Condition Code is set low (CC=1). If the record addressed by register 12 is to precede, the Condition Code is set high (CC=2). If the sequence of the two records in the sorted output is arbitrary, the Condition Code is set equal (CC=0).

The record sequence own code routine must return control to the Sort subroutine through the address saved in register 14. Registers 11 and 12 must still contain the addresses of the two records that were compared. The Condition Code is set to indicate the result of the comparison.

For variable-length records, registers 11 and 12 indicate the addresses of the first bin of each record. The four-byte record length field is considered to be part of the record (bin).

3.1.2. Data Reduction Own Code Routine

The user can process records having equal key fields with a data reduction own code routine. The address of the routine may be entered into the parameter table by means of the DROC keyword parameter. (This requires the DROC=symbol form of the keyword parameter.) Each time the Sort subroutine determines that two records have equal keys, either from its comparison routine or from the record sequence own code routine, it transfers control to the address specified by the DROC keyword parameter. Registers 11 and 12 contain the addresses of the two records. Register 14 contains the Sort subroutine return address. Register 15 contains the address of the data reduction own code routine.

The two records with equal keys may be combined by the user own code routine to form a single record. The record addressed by register 12 is eliminated, and the record addressed by register 11 is retained. The data reduction own code routine makes a single record, addressed by register 11, available to the Sort subroutine by incrementing by 4 the return address contained in register 14 and transferring control to the Sort subroutine at that address. If the user wishes to return both records to the Sort subroutine, control is returned to the address contained in register 14.

The user can eliminate records having equal key fields by specifying the DROC=DELETE keyword parameter. The Sort subroutine then automatically performs data reduction by arbitrarily eliminating either one of the two equal key records. No transfer of control to a user program routine is executed.

For variable-length records the four-byte record length field is considered part of the record. The record length fields of both records must not be altered by the problem program. Variable-length records to be processed by the data reduction own code routine are contained in contiguous storage and are not separated into bins.

The data reduction own code routine must not alter the contents of registers 11, 12, or 14 or the key fields of the records to be retained. The record length field for variable-length records may not be altered.

4. LINKAGE BETWEEN PROBLEM PROGRAM AND SORT SUBROUTINE

4.1. GENERAL

The linkage or interface between the problem program and the Sort subroutine may be accomplished by various macro instructions. This section describes the problem program linkages required to utilize the Sort subroutine for small volume sort runs. Additions to, or subsets of, these linkages are required for large volume sorts and are described in Section 5.

The Sort subroutine linkages may be written in Assembler source code language, or linkages may be generated by means of macro definitions. Both methods of generation are described for each linkage. When macro instructions are used, the system STDEQU macro instruction must be in effect. Registers not used directly for linkage communication are saved and restored by the Sort subroutine. Condition Code status is not saved.

4.2. SORT CALL – MR$ORT

All communication between the problem program and the Sort subroutine is controlled by an interface module, MR$ORT. This module is part of the problem program, must reside in main storage before the Sort subroutine is initialized, and must remain in main storage during the sorting process. The interface module may be linked into the problem program by defining MR$ORT as an external reference (EXTRN). The problem program must then be processed by the Linkage Editor before it can be executed.

The interface module may also be assembled into the program by calling the MR$ORT macro instruction. (The sort module MR$ORT accesses the job preamble and System Information Block. Therefore, if the MR$ORT module is assembled with the problem program, the STDEQU macro instruction must contain the JP and SB parameters. See *UNIVAC 9400 Supervisor Programmers Reference, UP-7689* (current version).) The module is then incorporated inline in the problem program. The MR$ORT macro definition has the following format:

LABEL	OPERATION	OPERAND
[symbol]	MR$ORT	

When the MR$ORT macro instruction is assembled, an Assembler DROP statement is generated for registers 0, 1, 2, 3, 4, 5, 14, and 15. If any of these registers is required by the problem program, USING Assembler statements for each register should be issued following the MR$ORT macro instruction.

4.3. INITIALIZATION – MR$OPN

Before any records are delivered to the Sort subroutine for processing, the initialization linkage coding or the MR$OPN macro instruction must be executed. Register 1 must contain the address of the Sort parameter table. The Sort subroutine returns control to the problem program at the address specified by the IN keyword parameter. Sort initialization performs all necessary functions required for this particular sort run as determined from the Sort parameter or from Sort control stream parameters.

The use of the MR$OPN macro instruction when executing or resuming a large volume sort is described in Sections 5 and 6.

The user initializes the Sort subroutine by using either source code language or the MR$OPN macro instruction.

- Source Code

  The following source code provides the necessary linkage to initialize the Sort subroutine.

LABEL (1)	OPERATION (10)	OPERAND (16)
[symbol]	L	RF$,=A(MR$ORT)
	XR	R0$,R0$
	BALR	RE$,RF$

  The problem program is responsible for loading register 1 with the address of the Sort parameter table.

- Macro Instruction

  The MR$OPN macro instruction may be used to generate the Sort initialization linkage. The Sort subroutine loads register 1 with the address of the parameter table, if specified. If the address is not specified, the problem program must load register 1 with the address of the parameter table. The format of the MR$OPN macro instruction is:

LABEL	OPERATION	OPERAND
[symbol]	MR$OPN	[param-table-name]

where:

param-table-name is the symbolic label of the MR$PRM macro instruction. If param-table-name is not specified, a comma/blank must be specified if comments are specified.

4.4. RECORD RELEASE – MR$REL

The problem program must provide linkage to release each record to the Sort subroutine, one record at a time. The MR$REL macro instruction or the record release linkage coding must be executed. Register 1 must contain the address of the first byte of the record to be sorted. The Sort subroutine returns control to the problem program at the instruction immediately following the record release linkage. If variable-length records are being sorted, the first four bytes of the record must contain the record length field.

The user releases a record to the Sort subroutine, one record at a time, by using either source code language or the MR$REL macro instruction.

- Source Code

  The following source code provides the linkage necessary to release a record to the Sort subroutine.

LABEL	OPERATION	OPERAND
[symbol]	L	RF$,=A(MR$OUT)
	LA	R0$,4(0,0)
	BALR	RE$,RF$

  The problem program is responsible for loading register 1 with the address of the first byte of the record to be sorted.

- Macro Instruction

  The MR$REL macro instruction may be used to generate the linkage to release a record. The format of the MR$REL macro instruction is:

LABEL	OPERATION	OPERAND
[symbol]	MR$REL	

4.5. SORT COMMAND – MR$SRT

The problem program must notify the Sort subroutine that all records to be sorted have been released. The Sort subroutine can then complete the sorting process. The Sort subroutine returns control to the problem program at the address specified by the OUT keyword parameter. The user notifies the Sort subroutine that all records are released by using either source code language or the MR$SRT macro instruction.

- Source Code

  The following source code provides the linkage that informs the Sort subroutine that all records have been released.

```
LABEL      OPERATION      OPERAND
1          10             16
[symbol]   L              RF$,=A(MR$OUT)
           LA             R0,$,8(0,0)
           BALR           RE$,RF$
```

- Macro Instruction

  The MR$SRT macro instruction may be used to generate the linkage to notify the Sort subroutine that all records have been released. The format is:

LABEL	OPERATION	OPERAND
[symbol]	MR$SRT	

4.6. RETURN SORTED RECORD – MR$RET

After all records have been sorted and the Sort subroutine has transferred control to the problem program at the address specified by the OUT keyword parameter, the problem program requests the return of a sorted record from the Sort subroutine.

The address of the first byte of the record being returned is loaded into register 1 by the Sort subroutine and control is transferred to the problem program at the instruction immediately following the record return linkage. For variable-length records, the address in register 1 is that of the four-byte record length field.

When all records have been returned to the problem program, the Sort subroutine transfers control to the problem program at the address specified by the FIN keyword parameter.

The user requests the return of a sorted record, one record at a time, by using either source code language or the MR$RET macro instruction.

- Source Code

  The following source code provides the linkage necessary to return a sorted record to the problem program.

```
LABEL      OPERATION      OPERAND
1          10             16
[symbol]   L              R1,F$,=A(MR$ORT)
           LA             R0,$,12(0,0)
           BALR           RE,$,RF$
```

- Macro Instruction

  The MR$RET macro instruction may be used to generate the linkage to return a sorted record. The format is:

LABEL	OPERATION	OPERAND
[symbol]	MR$RET	

5. LARGE VOLUME SORTS

5.1. GENERAL

A large volume sort is executed in three parts. These parts can be executed either automatically, with the Sort subroutine controlling the entire operation, or nonautomatically, one part at a time. Operator intervention is required to load, unload, and label tapes for either method of operation. The three parts of a large volume sort are:

- Part A – The production of intermediate files stored on magnetic tape.
- Part M – The merging of Part A output files into larger intermediate output files.
- Part F – The final merge of output files from Part M. If the volume of data is not too large, Part M may be bypassed and Part F will be the final merge of output files from Part A.

For both automatic and nonautomatic large volume sorts, the standard sort filenames (SM01, SM02,..., SM14) designate the intermediate sort output files. Two tape units may be assigned to this sort filename by using the ALT option of the DVC Job Control statement at the time the devices are assigned to the sort run. After a reel is written, the tape is rewound with interlock, console messages are supplied to the operator, and a tape swap is performed. For Part M and Part F processing, each intermediate input file may also be assigned two tape units.

This section describes both automatic and nonautomatic methods of operation, the respective parameters, and the linkages available to the problem program.

5.2. AUTOMATIC LARGE VOLUME SORTS

The Sort subroutine provides for the automatic execution of all three parts of a large volume sort; upon completion of one part, it automatically executes the next part. The Sort subroutine controls the operation of the sort and provides console messages for operator action.

The AUTO keyword parameter must be included in the Sort parameter table (see 2.3.16). Three specifications, one of which is optional, are allowed with this keyword parameter. The two-character user identification and the standard sort filename are required. The maximum number of records to be sorted in a cycle is an optional specification.

The linkages described in Section 4 are required for an automatic large volume sort. The Sort parameter table (MR$PRM) and the Sort interface module (MR$ORT) are also required. Two optional linkages may be used: one to terminate a Part A cycle and one to allow resumption of Part F of the sort.

5.2.1. Part A Cycle Break – MR$BRK

The user may wish to terminate or break cycles at particular points during Part A processing of automatic large volume sorts. When the cycle break linkage is issued, the Sort subroutine completes sorting the records released to it, produces the Part A intermediate output file, and returns control to the problem program at the instruction following the cycle break linkage coding.

The user may request a cycle break by using either source code language or the MR$BRK macro instruction.

- Source Code

The following source code provides the linkage that terminates a cycle at a given point during Part A processing of a large volume sort.

```
LABEL      OPERATION    OPERAND
1          10           16
[symbol]   L            RF$,=A(MR$ORT)
           LA           R0$,16(0,0)
           BALR         RE$,RF$
```

- Macro Instruction

The MR$BRK macro instruction may be used to generate the linkage that terminates a cycle during Part A processing. The format is:

LABEL	OPERATION	OPERAND
[symbol]	MR$BRK	

5.2.2. Request for Rerun Information – MR$REC

The Sort subroutine maintains a table with rerun information while executing Part F of a large volume sort. This table is required in order to resume an interrupted sort from some point in Part F (see 6.5). The Sort subroutine returns control to the problem program at the instruction following the linkage requesting the rerun information. Register 1 contains the address of the first byte of the rerun information table and register 0 contains the length, in bytes, of the table. If the rerun information table is not available when MR$REC is executed, registers 0 and 1 are set to binary zero values. The user cannot resume Part F for this condition.

The user requests the rerun information table in order to resume an interrupted Part F sort by using either source code language or the MR$REC macro instruction.

- Source Code

  The following source code provides the linkage that supplies the address of the first byte and length of the Sort rerun information table.

```
LABEL      OPERATION      OPERAND
1          10             16
[symbol]   L              RF$,=A(MR$OR T)
           LA             R0$,20(0,0)
           BALR           RE$,RF$
```

- Macro Instruction

  The MR$REC may be used to generate the linkage that supplies the address of the first byte and length of the Sort rerun information table. The format is:

LABEL	OPERATION	OPERAND
[symbol]	MR$REC	

5.3. NONAUTOMATIC LARGE VOLUME SORTS

A large volume sort may be executed nonautomatically as three separate parts: Part A, Part M, and Part F. The specific part is designated by the DOA, DOM, or DOF keyword parameter, respectively.

5.3.1. Execution of Part A

The execution of Part A of a nonautomatic large volume sort is performed if the DOA keyword parameter is included in the Sort parameter table. (See 2.3.18 for the required and optional specifications for the DOA keyword parameter.) The Sort subroutine linkages MR$OPN, MR$REL, MR$SRT, and MR$BRK are used as described in 4.3, 4.4, 4.5, and 5.2.1, respectively. After all records have been sorted (MR$SRT) and upon completion of the final cycle of Part A, the Sort subroutine returns control to the problem program at the address specified by the DOEXT keyword parameter. If the DOEXT keyword parameter is not specified, the Sort subroutine transfers control to the system end-of-job procedure which terminates the run.

5.3.2. Execution of Part M

The execution of Part M of a nonautomatic large volume sort is performed if the DOM keyword parameter is included in the Sort parameter table. (See 2.3.19 for the specifications for the DOM keyword parameter.) The Sort subroutine linkage MR$OPN is required.

When the MR$OPN linkage is executed, the Sort subroutine immediately enters Part M processing and produces a single intermediate output file by merging the input files specified by the DOM keyword parameter. The maximum number of input files is one less than the actual number of sort files assigned to the sort by Job Control. All intermediate input files merged must begin with reel 1 of the file. The execution of Part M is required only if the number of intermediate files is greater than the number of sort files that can be assigned to the Part F run. Part M must be executed repetitively until the number of intermediate files is less than, or equal to, the numbe of sort files assigned to the Part F run.

When the execution of Part M is completed, the Sort subroutine transfers control to the problem program at the address specified by the DOEXT keyword parameter. If the DOEXT keyword parameter is not specified, the Sort subroutine transfers control to the system end-of-job procedure which terminates the run.

5.3.3. Execution of Part F

The execution of Part F of a nonautomatic large volume sort is performed if the DOF keyword parameter is included in the Sort parameter table. (See 2.3.20 for the specifications for the DOF keyword parameter.) The Sort subroutine linkages MR$OPN and MR$RET are required; MR$REC is optional. When the MR$OPN linkage is executed, the Sort subroutine immediately enters Part F processing. This processing performs the final merge of the intermediate files while returning the sorted records to the problem program.

The Sort subroutine returns control to the problem program at the address specified by the OUT keyword parameter. The problem program may then request the return of all sorted records by using the MR$RET linkage. When the Sort subroutine has returned all records to the problem program, it transfers control to the problem program at the address specified by the FIN keyword parameter.

The number of intermediate files that Part F processing accepts is equal to the number of standard sort filenames assigned by Job Control. The maximum number of files is 14.

6. RERUN FACILITIES

6.1. GENERAL

The Sort subroutine requires the restoration of all facilities assigned to it in order to resume an interrupted sort. Because the Sort subroutine is incorporated into the problem program, it has no control of facilities required by the problem program. Therefore, if an interrupted sort is to be continued, the problem program is responsible for restoring those facilities under its control to their rerun condition.

An interrupted sort may be continued or the output of a specific Sort Part may be re-created by adding the proper rerun keyword parameter to the original set of parameters prepared for the sort.

The Sort subroutine has three rerun provisions:

- Resumption from a tape collation point;
- Resumption from a specific part of a large volume sort;
- Re-creation of the output of a specific part of a large volume sort.

This section describes resuming an interrupted sort from a specific point and re-creating a specific output of Part A or Part M. The specifications to resume a sort are applicable to either automatic or nonautomatic large volume sorts. If specified, DROC and RSOC own code routines in the problem program are not executed until the point of resumption is reached.

6.2. RESUMPTION FROM A TAPE COLLATION POINT

A small or large volume sort may be resumed from a collation point. During tape collation processing, the Sort subroutine writes the necessary rerun information onto a work tape prior to each collation pass. A console message informs the operator of the recovery number and the tape unit that contains the rerun information. As each tape collation pass is satisfactorily completed, the information required for a rerun is destroyed. Consequently, the Sort subroutine may be resumed only from the collation pass that was interrupted, that is, from the most recent recovery number supplied to the operator.

An interrupted sort may be resumed from a tape collation point by adding the RESUME=(PASS,...) keyword parameter to the original set of parameters prepared for the sort and then executing the original program. (See 2.3.21 for the specifications for the RESUME keyword parameter.) The scratch tapes must be assigned and must contain the same data as they did at the point of interruption. When the MR$OPN initialization linkage is executed, the RESUME keyword parameter is detected. The recovery number given in the console message is used to access the sort rerun information; the collation tapes on the tape units are validated and repositioned. The Sort subroutine then returns to the address specified by the IN keyword parameter. (The problem program is re-executed.) The Sort subroutine accepts and bypasses all records released to it up to the point of interruption. At this time, the collation phase is completed and normal processing is reinstated.

6.3. RESUMPTION OF PART A

During the execution of Part A of a large volume sort, console messages supply the operator with the label, cycle number, and the beginning and ending record numbers of each intermediate file produced. A large volume sort interrupted during Part A may be resumed by adding the RESUME=(CYCLE,...) keyword parameter to the original set of parameters prepared for the sort and then executing the original program. (See 2.3.21 for the specifications for the RESUME keyword parameter.)

When the MR$OPN initialization linkage is executed, the RESUME keyword parameter is detected. The Sort subroutine returns control to the problem program at the address specified by the IN keyword parameter. The problem program issues the record release linkage (MR$REL) as though the program were being re-executed from the beginning. The Sort subroutine bypasses all records released to it until the first record for the cycle and the record number specified in the RESUME keyword parameter are received.

The Sort subroutine maintains the current cycle number and record count as two contiguous fields in storage. The address of these two fields is stored in the MR$ORT module at the location labeled MR$ADR. If the MR$ORT interface module is linked with the problem program, by means of the Linkage Editor, MR$ADR should be defined as an external reference (EXTRN).

The cycle number and record count may be accessed by the problem program in order to coordinate problem program restart procedures with the resumption of an interrupted sort. At convenient points within the problem program, the user can establish restart checkpoints which include the current cycle number and record count retrieved by means of the address specified in MR$ADR. If the run is interrupted and requires a subsequent RESUME or REDO, the run can then be restarted from the user checkpoint. After the MR$OPN initialization linkage is executed, which processes the RESUME keyword parameter, the cycle number and record count can be updated in the locations specified by the address in MR$ADR. Thus, only the records from the restart point to the cycle and the record numbers specified by the RESUME keyword parameter need to be released to, and bypassed by, the Sort subroutine.

6.4. RESUMPTION OF PART M

During the execution of Part M of a large volume sort, console messages supply the operator with the label, reel number, cycle number, and the beginning and ending record numbers contained in each output reel produced by Part M processing. Execution of Part M may be resumed by adding the RESUME=(MERGE,...) keyword parameter to the original set of parameters prepared for the sort and then executing the original program. (See 2.3.21 for the specifications for the RESUME keyword parameter.)

When the MR$OPN initialization linkage is executed, the RESUME keyword parameter is detected. The Sort subroutine requests that the reel containing the rerun information and the associated intermediate files be loaded. After the Part M rerun information is processed, the Sort subroutine directly enters Part M processing and continues merging the files from the point of interruption.

A sort resume information block is written at the beginning of each output reel produced by Part M. When RESUME=(MERGE, tape-label, reel-number) is specified, the resume information block of the tape-label and reel-number is read; the interrupted Part M sort is resumed from this point. The specified RESUME tape is then rewritten.

6.5. RESUMPTION OF PART F

During the execution of Part F of a large volume sort, the Sort subroutine maintains a rerun information table. This table is required in order to resume a sort interrupted at some point in Part F. The user requests the address and length of this table by means of the MR$REC request linkage (see 5.2.2). The Sort subroutine returns the address in register 1 and the length in register 0. This rerun information must be saved by the problem program. If registers 0 and 1 contain the binary value zero after an MR$REC execution, a Part F resumption is not established.

Execution of Part F may be resumed by adding the RESUME=(FINAL,symbol) keyword parameter to the original set of parameters prepared for the sort. The specification "symbol" is the starting address of the rerun information table that is given to, and saved by, the problem program when the MR$REC request linkage is executed.

6.6. RE-CREATION OF A PART A FILE

If an output reel produced during Part A processing cannot be read, the output file incorporating this reel can be re-created. The entire file must be re-created because the sorted file was created from random input data. Re-creation of a Part A output file operates in the same manner as the resumption of Part A except that the run is terminated after the specific file has been sorted.

A Part A output file may be re-created by adding the REDO=(CYCLE,...) keyword parameter to the original set of parameters prepared for the sort. (See 2.3.22 for the specifications for the REDO keyword parameter.) These specifications are supplied to the operator by means of console messages when Part A was originally executed.

The Sort subroutine detects the REDO keyword parameter when the MR$OPN initialization linkage is executed. The Sort subroutine returns control to the address specified by the IN keyword parameter (see 2.3.7). It then accepts records from the problem program and bypasses all records until the specified "from" record number is reached.

Records are then accepted and processed until the specified "to" record number is reached. The entire group of records is sorted and the intermediate file is produced. The Sort subroutine terminates the run by means of the system end-of-job procedure.

6.7. RE-CREATION OF A PART M TAPE

An intermediate output tape produced during Part M of a large volume sort may be re-created. Re-creation of a Part M output tape operates in the same manner as the resumption of Part M except that the run is terminated after the tape has been re-created.

A Part M output tape may be re-created by adding the REDO=(MERGE,...) keyword parameter to the original set of parameters prepared for the sort. (See 2.3.22 for the specifications for the REDO keyword parameter.)

The Sort subroutine detects the REDO keyword parameter when the MR$OPN initialization linkage is executed. The Sort subroutine requests that the input tapes required to re-create the desired output tape be loaded. The labels are validated, the tapes are positioned, and the specified output tape is re-created. The Sort subroutine then terminates the run by means of the system end-of-job procedure.

A sort resume/redo information block is written at the beginning of each output reel produced by Part M. This block must be readable in order to re-create a Part M output reel.

7. CONTROL STREAM PARAMETERS

7.1. GENERAL

The ability to process certain sort parameters submitted at execution time is provided by the Sort subroutine. The sort parameters accepted and processed from a program control stream are:

TAPES	AUTO	BIN
DISC	DOA	NOCKSM
SHARE	DOM	RESUME*
RESERV	DOF	REDO

* RESUME = (FINAL, ...) is not accepted.

Control stream parameters facilitate the incorporation of optional information into the sort parameter table, the addition of new parameters, or the overriding of parameters already present in the parameter table. Sort parameters submitted through the control stream take precedence over the same parameters which may be present on the parameter table supplied to the sort by means of the MR$OPN macro instruction.

7.2. CONTROL STREAM PARAMETER SPECIFICATION

Sort parameters are entered into the sort by means of the // PARAM Job Control statement. The parameters are written in columns 10 through 71 of the PARAM statement; the parameters must begin in column 10 of each PARAM statement submitted; parameters are written in the same keyword format as described for the MR$PRM macro instruction. A space following a parameter terminates the PARAM sort parameters. Parameters are separated by a comma. Parameters may be continued on more than one PARAM statement by writing parameters either contiguously in columns 10 through 71, or by following a parameter with a comma and space.

Example:

```
//PARAM TAPES=(STD, 6), SHARE=SM01,
//PARAM NOCKSM=T
```

7.3. CONTROL STREAM PARAMETER PROCESSING

If the parameter CSPRAM = NO is specified or is optionally indicated as NO by the operator, control stream parameter processing is bypassed.

If the parameter CSPRAM = YES or is optionally indicated as YES by the operator, the user program's control stream is accessed for sort PARAM statements. Parameters are accepted, validated, and processed until a space following a parameter is detected.

Control stream processing is instituted during sort initialization; that is, when the MR$OPN macro instruction is executed. The user is responsible for the correct placement of sort PARAM statements in the control stream. Caution must be exercised as to the relative placement of sort PARAM statements and the execution of the MR$OPN macro instruction.

APPENDIX A. PREPARATION OF A SMALL VOLUME SORT

The user incorporates the UNIVAC 9400 Sort/Merge subroutine into the problem program. The problem program is responsible for supplying the unsorted records to the Sort subroutine, and for processing the sorted records returned by the Sort subroutine. Figure A–1 is a functional sort program block diagram.

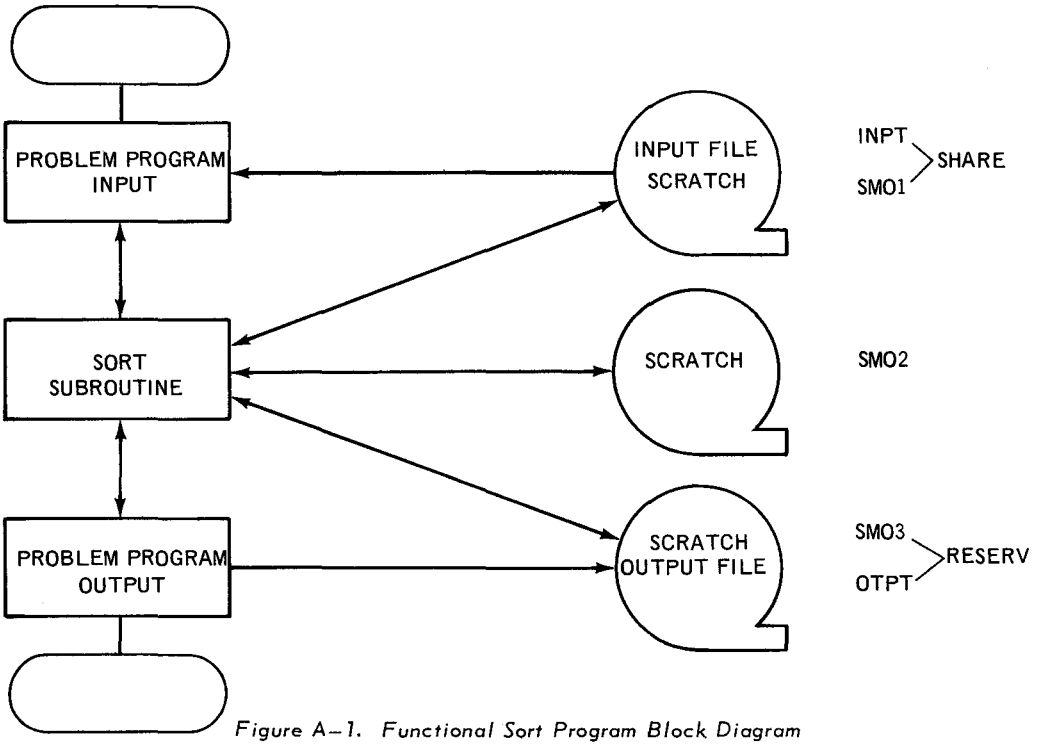

Figure A–1. Functional Sort Program Block Diagram

The following sample program is an example of a basic small volume sort problem program. Data Management is used to process the input and output records. The Linkage Editor is used to link the user program with related sort and Data Management elements. The sample program assumes a fixed block size of 400 bytes and a fixed record size of 80 bytes. The records are to be sorted on two key fields in ascending sequence.

The major key is in character format, and begins in record position 10 and is 6 bytes long. The minor key is in packed decimal format, begins in record position 77, and is 4 bytes long. This sample program illustrates a two step job consisting of a tape assembly and a Tape Linkage Editor run.

The program sample illustrates the use of the Sort subroutine. Many variations are applicable in the sort description, use of Data Management, and in record processing. If disc storage is available, the same program could be modified to a tape/disc sort by adding a DISC keyword parameter to the sort description (by means of MR$PRM or control stream parameters). For a disc only sort, the sample program would be modified to utilize disc Data Management calls. The TAPES, SHARE, and RESERV keyword parameters are removed and a DISC keyword parameter is added to the sort parameter table.

```
// JOB ASMRUN
// DVC 3
// LFD PRNTR
// DVC 5
// VOL SYSRES
// LFD SYSRES
// DVC 6
// LFD SCR1
// DVC 7
// LFD SCR2
// DVC 8
// LFD OBJFIL
// EXEC ASM
/$
PR32     START 0


         STDEQU
INPT     DTFMT  BKSZ=400,RCSZ=80,FLBL=STD,RCFM=FIXBLK,IORG=8,          X
                EOFA=EOF,IOA1=IN1,IOA2=IN2
OTPT     DTFMT  BKSZ=400,RCSZ=80,FLBL=NO,RCFM=FIXBLK,CLRW=RWD,IORG=9,  X
                IOA1=OUT1,IOA2=OUT2,TYPE=OUTPUT
         EXTRN  MR$ORT                                                 NOTE 1
         ENTRY  BEGIN
BEGIN    BALR   2,0
         USING  *,2
         L      13,=A(AREA9400)                                        NOTE 2
         B      START
         DS     OF
AREA9400 DC     XL72'0'                                                NOTE 2
SORT     MR$PRM  RCSZ=80,IN=C001,OUT=C002,FIN=BND,                     X  NOTE 3
                STOR=LAST,FIELD=(9,6,CH,A,1,76,4,PD,A,2)
IN1      DS     5CL80                  INPUT AREA 1
IN2      DS     5CL80                  INPUT AREA 2
OUT1     DS     5CL80                  OUTPUT AREA 1
OUT2     DS     5CL80                  OUTPUT AREA 2
START    OPEN   INPT
         MR$OPN SORT                                                   NOTE 4
C001     GET    INPT
         LR     1,8
         MR$REL                                                        NOTE 5
         B      C001
EOF      CLOSE  INPT
         MR$SRT                                                        NOTE 5
```

```
C002     OPEN   OTPT
C003     MR$RET                                          NOTE 5
         MVC    0(80,9),0(1)
         PUT    OTPT
         B      C003
BND      CLOSE  OTPT
         EOJ
         LTORG
LAST     EQU    *
         END    BEGIN
/*
// DVC 7
//       LFD    LDMFIL                                   NOTE 6
// EXEC  LINK
/$
         LOADM PR32,X'0'
         INCLUDE PR32,*                                  NOTE 7
         ENTER BEGIN
/*
/&
```

NOTE 1 – Causes the sort control routine, MR$ORT, which calls in and relocates the sort modules from SYSRES to be linked into the problem program.

NOTE 2 – Data Management requires a 72 byte save area beginning on a fullword boundary. The address of this area must be loaded into register 13 before the Data Management routines can be accessed.

NOTE 3 – MR$PRM generates the sort parameter table from the specified parameters.

RCSZ=80 specifies the record size.

IN=C001 defines the address to which control is returned in the problem program after the Sort subroutine has been initialized as a result of MR$OPN.

OUT=C002 specifies the address to which control is returned in the problem program after the Sort subroutine has completed the collation passes of records following the execution of MR$SRT.

FIN=BND defines the address to which control is transferred in the problem program after all sorted records have been delivered to the problem program.

STOR=LAST indicates the main storage address at which the sort control routine can load and execute the required sort modules. The balance of available storage is used for the sort tournament area.

NOTE: When the Automatic Include function of the Linkage Editor is invoked, the linked I/O modules are linked into the program preceding the sort module. As a result, LAST is the highest storage location +1 of the linked program.

FIELDS=(9,6,CH,A,1,76,4,PD,A,2) describes the key fields according to which each record is to be sorted. (See Section 2 for description of parameters.)

NOTE 4 – MR$OPN causes the sort initialization linkage to be generated. Register 1 is loaded with the address of the sort parameter table.

NOTE 5 – MR$REL releases records to the sort, one at a time. The records are in turn distributed to work tapes SM02 and SM03. The sort returns control to the problem program at the instruction immediately following MR$REL (B C001). The program remains in this loop until the end of the input file is reached. The input file is then closed and MR$SRT is executed indicating that all input records have been released to the sort. The user is given the opportunity to replace the input tape with a scratch tape. The sorting process then proceeds until the final merge is to be performed from SM01 and SM02 to the output tape. At this point, the sort returns to the problem program at the address specified by the OUT parameter (C002) which opens the output file. MR$RET causes sorted records to be returned to the problem program one at a time with the address of the next sequential record in Register 1. After all sorted records have been returned to the problem program, the sort transfers control to the program at the address specified by the FIN keyword parameter (BND).

NOTE 6 – The Linkage Editor process is the second step of this two step job. The device and file assignments from step one are adequate for step two with one exception. We must assign a tape device for the creation of the linked load module. Logical device 7 is assigned to this function (// DVC 7 and // LFD LDMFIL). The input is on logical device 8 which is the object module (OBJFIL) produced by the Assembler.

NOTE 7 – The Automatic Include function will cause the necessary Data Management modules to be linked into the output Load Module along with the object modules produced by the Assembler. MR$ORT is also linked into the problem program.

The following control stream will affect execution of the linked sort program:

```
// JOB PR32
// DVC 5
// VOL SYSRES
// LFD SYSRES
// DVC 6
// LFD INPT
// DVC 6
// LFD SM01
// DVC 7
// LFD SM02
// DVC 8
// LFD OTPT
// DVC 8
// LFD SM03
// EXEC PR32,,SM02
// PARAM TAPES=(NO,3)
// PARAM SHARE=SM01,
// PARAM RESERV=SM03
/$
/*
/&
```

NOTE 1

NOTE 2

NOTE 1 – The double definition of device 6 (INPT and SM01) and device 8 (OTPT and SM03). This provides the necessary file definitions for both the sort and Data Management. Note also that the linked sort module is being loaded from device 7 which contains the output Load Module from the Linkage Editor. This tape should be replaced by a scratch tape (SM02) after the sort has been loaded.

NOTE 2 – The sample program sort MR$PRM macro instruction does not specify a CSPRAM keyword, consequently, at execution of the sample program CSPRAM=OPTION is assumed by the sort and the console message SM01 033 SORT // PARAM is issued. An operator response of Y (yes) is required. To eliminate this console message and operator response, the CSPRAM=YES keyword parameter should be included in the MR$PRM sort macro instruction.

TAPES=(NO,3) specifies that three tapes are to be used by the Sort subroutine for work tapes and that NO labels exist on the assigned tape files. The two possible entries for this keyword parameter are STD (standard) and NO.

// PARAM SHARE=SM01 specifies that the input file will be mounted on logical device 6 (DVC 6) during phase 1 (dispersion phase) of the sort. This tape unit is not used by the sort until all records have been released by the problem program. This tape unit is then rewound and a request is issued to load a scratch tape (SM01).

// PARAM RESERV=SM03 specifies that logical device 8 is to be used by the problem program for its output after the sort's collation phases. As a result, the sort will not use SM03 during the final merge, but it will be available for problem program output.

APPENDIX B. SORT FACILITY REQUIREMENTS

B.1. STORAGE REQUIREMENTS

The minimum work area made available to the sort by the problem program by means of the STOR parameter must be at least 6600 bytes.

$$((\text{address} + \text{bytes})/8) * 8 - ((\text{address} + 7)/8) * 8 \geq 6600 \text{ bytes}$$

address – STOR address

bytes – either specified as STOR number-of-bytes parameter, or obtained as the program upper bound minus address

The maximum area required by the sort is dependent on record size, number of tapes, disc area available, and type of sort (small or large volume). The maximum working storage possible should always be assigned to the sort.

MR$ORT – 350 bytes

linkages – see Sections 4 and 5. Note linkages generate literals.

MR$PRM – dependent on parameters specified. See sort parameter descriptions (Section 2) to determine the amount of storage required.

B.2. MAGNETIC TAPE REQUIREMENTS

Magnetic tapes are assigned by means of sort filenames SM01 through SM14. A minimum of three filenames (SM01, SM02, and SM03) and three tape units must be assigned for a small or large volume tape or disc/tape sort.

For operational phases 1, 2, and 3a of a single cycle sort, up to six filenames may be assigned (one tape unit per filename) and will be utilized for sorting.

Up to six filenames may be assigned and utilized for sorting Part A of a multicycle sort, however, an alternate tape unit may also be assigned to the Part A intermediate output cycle, resulting in six filenames and seven tape units for the sort.

Up to 14 filenames (SM01 through SM14) can be assigned for operational phases 4 and 5 (see Section 1). Alternate tape units may be assigned for the output sort files of phases 3b and 4. Alternate tape units are also processed (may be assigned) for input sort files of phases 4 and 5.

If tapes have standard labels (TAPES=STD), the sort bypasses VOL labels. The HDR1 label is rewritten with the following fields updated; all other HDR1 fields are retained.

identification field – CL17 'SORT SCRATCH LABL'

expiration date – current system (SIB) date

creation date — current system (SIB) date

The maximum tape block written is 4095 bytes; the minimum block written is 18 bytes.

B.3. DISC REQUIREMENTS

The disc storage space for each disc unit assigned to the sort is obtained from the Supervisor. The largest contiguous system scratch area of each disc is used.

When disc space is allocated, an internal logical buffer and a physical input/output block size are computed for a 2-way disc merge. Storage must accommodate three buffers and two output blocks. Ideally, storage available for buffer/block usage should allow for full track capacity (5*3625 plus sort code which can range from 4000 to 6000 bytes). If storage permits, the order of merge is increased by the number of additional buffers which can be accommodated above the minimum requirements.

When possible, the track capacity logical buffer/block sizes are used by the disc sort. The total disc storage capacity is reduced by the following:

- Sort control information required in each block (23 bytes)
- Physical reduction in track capacity for more than one block per track. If more than one block per track is written, n tracks are reserved from the total capacity for use by the disc merge.
- Strings are not bridged across tracks, that is, the last block of a string can be a partial track.
- Record type waste:

  fixed length records – the block size is reduced to an integral number of records.

  variable length records – the record mix causes the average block size to be less than the maximum block size.

The minimum sort disc storage capacity utilized by the sort must be large enough to hold 2 strings – or disc storage will not be used. The minimum disc capacity required can be approximated as:

$$4* (M-5000-2* (R*(3625/R)))$$

where M is the storage assigned to the sort and R is the record size.

B.4. DISC SORT/TAPE SORT RELATIONSHIP

Disc and tape units may be specified for a sort in the UNIVAC 9400 System. When both auxiliary storage types are specified, the Disc Sort is used to increase the string length for the Tape Sort. Disc Sort cycles are executed in which Disc Final passes a single string to the Tape Accept module. Disc Sort/Tape accept cycles (operational phase 1) are terminated when either the tape capacity is reached or there are no more records to be sorted. When the acceptance phase is terminated, the tape polyphase (operational phase 2) and/or Tape Final (operational phase 3) modules are executed.

The use of disc and tape result in the following sort conditions:

- The disc capacity utilized by the sort is limited to the capacity of one reel of tape.
- If disc capacity (DCAP) exceeds tape capacity (TCAP) divided by the number of sorting tapes (t), up to six, then the tape polyphase (operational phase 2) module is not executed. Tape Accept (operational phase 1) distributes one string to each available sort tape; each string is the length of DCAP. The Tape Final merge (operational phase 3) is then executed.

  When this condition exists, the total capacity of a sort cycle is increased to DCAP*t, where t is the number of available sort tapes.
- If DCAP<(TCAP/t) then the capacity of a sort cycle is limited to one tape reel. The tape polyphase (operational phase 2) module is executed if required.

APPENDIX C. TAPE LABELING FOR LARGE VOLUME AUTOMATIC SORTS

During Part A of a large volume sort, a number of intermediate output files are produced, each file consisting of one or more reels. As each reel is completed, information typed on the console typewriter instructs the operator to dismount and label the reel. When all input records have been delivered to the sort and all intermediate output files have been produced in Part A, one or more merge runs are required, depending on the number of intermediate files and the number of tape units available for merging. If the number of tape units exceeds or is equal to the number of files, only one merge run is necessary and the routine proceeds directly from Part A to Part F. During Part F ordered data is returned to the problem program one record at a time.

If the number of files to be merged exceeds the number of tape units available, one or more merge runs is required before the final merge. These intermediate merges constitute Part M of the automatic large volume sort. As in Part A, the program types out instructions for labeling and mounting tape files.

The labels used for output files during both Part A and Part M consist of the following format:

uu/ccc

where:

uu is the two-character label code specified by the user.

ccc is the output file cycle number. For an AUTO or DOA sort, ccc is produced by the sort; all output files are numbered in ascending sequence, that is, uu/001, uu/002, ... For a DOM sort, ccc is specified by the user.

All merging in Part M, with the possible exception of the first merge, is done using all assigned tape units. If necessary, the first merge reduces the total number of files to a point where all tape units assigned can be used for each subsequent merge.

An alternate tape device may be assigned for Part A and Part M intermediate sort *output* files. Alternate tape devices may be assigned for Part M and Part F intermediate sort *input* files.

The following table illustrates the merging sequence and labels for an AUTO sort. The user label code is LC. Thirty-six files are produced by the Part A, that is, LC/001 through LC/036. Assume five tape units are assigned to the sort and assigned as DVC's 5, 6, 7, 8, and 9. All sort output is to tape DVC 9.

TAPE DVC	INPUT LABEL	OUTPUT LABEL	INPUT LABEL	OUTPUT LABEL
5	LC/036			
6	LC/035			
9		LC/037		
5	LC/034		LC/006	
6	LC/033		LC/005	
7	LC/032		LC/004	
8	LC/031		LC/003	
9		LC/038		LC/045
5	LC/030		LC/002	
6	LC/029		LC/001	
7	LC/028		LC/037	
8	LC/027		LC/038	
9		LC/039		LC/046
5	LC/026		LC/039	
6	LC/025		LC/040	
7	LC/024		LC/041	
8	LC/023		LC/042	
9		LC/040		LC/047
5	LC/022			
6	LC/021			
7	LC/020			
8	LC/019			
9		LC/041	LC/043	
5	LC/018		LC/044	Final Output
6	LC/017		LC/045	
7	LC/016		LC/046	Part F
8	LC/015		LC/047	
9		LC/042		
5	LC/014			
6	LC/013			
7	LC/012			
8	LC/011			
9		LC/043		
5	LC/010			
6	LC/009			
7	LC/008			
8	LC/007			
9		LC/044		

Table C–1. Merging Sequence and Labels for an AUTO Sort

APPENDIX D. KEYWORD PARAMETER REQUIREMENTS

The STOR and RCSZ keyword parameters are required for any sort. BIN, DROC, NOCKSM, CSPRAM, PAD, and ADTABL keyword parameters are optional for any sort. If the TAPES and DISC keyword parameters are omitted for a small volume sort, the sort is processed as an internal-only sort.

KEYWORD PARAMETERS	SMALL VOLUME	AUTO	DOA	DOM	DOF
IN	R	R	R		
OUT	R	R	R		
FIN	R	R			R
FIELD or RSOC	R	R	R	R	R
TAPES	O	R	R	R	R
TAPES and DISC	O	O	O		
DISC	O	O	O		
SHARE	O				
RESERV	O				
DOEXT			O	O	
RESUME =					
PASS	O	O	O		
CYCLE		O	O		
MERGE		O		O	
FINAL		O			O
REDO=					
CYCLE		O	O		
MERGE		O		O	

R – required parameter
O – optional parameter
Blank – not applicable

APPENDIX E. MR$PRM ASSEMBLY TIME ERROR MESSAGES

The MR$PRM macro instruction outputs Assembler PNOTE messages with an asterisk error code notation when parameter errors or inconsistencies are detected. Parameter errors should be corrected before executing the sort program. A valid assembler truncation error may occur for AUTO, DOA, RESUME=MERGE, REDO=CYCLE, or REDO=MERGE keyword parameters. The following PNOTE error messages originate from the MR$PRM sort macro. Parameter entries are generated regardless of the error; 'param' is the parameter specified in the MR$PRM call.

* RCSZ, param INCORRECT. 0<RCSZ<16384.
The maximum-bytes value of the RCSZ keyword parameter is less than 1 or greater than 16383.

* BIN, param VALUE ERROR OR INCONSISTENT WITH RCSZ.
The bytes or minimum-bytes value of the BIN keyword parameter is greater than RCSZ or the parameter value is less than 1 or greater than 16383.

* SIZE, param INCONSISTANT WITH MIN-BIN OR RCSZ.
A size-1 value is missing, less than min-bytes of the BIN keyword parameter, greater than RCSZ, less than 1, or greater than 16383.

* VOL, param NOT WITHIN LIMITS. −1<VOL<101.
A vol-1 value of the BIN keyword parameter must be in the range 0 through 100.

* MINIMUM PARAMETERS FOR FIELD NOT PRESENT.
The byte-pos-1 and lgth-1 FIELD keyword parameters are required.

* BYTE-POS. param NOT WITHIN LIMITS.
The byte-pos-1 value of the FIELD keyword parameter is not within the RCSZ or BIN size specification.

* LENGTH, param NOT WITHIN LIMITS OR MISSING.
The lgth-1 value of the FIELD keyword parameter is missing, less than 1, or greater than 256.

* FIELD EXCEEDS RCSZ OR BIN SIZE.
The key field exceeds the RCSZ or BIN size specification.

* FORM, param INCORRECT.
Form specification of the FIELD keyword parameter is invalid. (CH, BI, FI, PD, or ZD.)

* LENGTH, param INCORRECT FOR FIELD FORM.
The lgth-1 size of the FIELD keyword parameter exceeds the 16 bytes for a PD or ZD form-1 key field.

* SEQUENCE, param INCORRECT.
Seq-1 of the FIELD keyword parameter must be A or D.

* ORDER NUMBER REQUIRED IN ALL FIELD DESCRIPTIONS.
When specified, order-1 of the FIELD keyword parameter must be specified in *all* key field descriptions.

* ORDER NO. MUST BE OMITTED IN ALL FIELD DESCRIPTIONS.
When omitted, order-1 of the FIELD keyword parameter must be omitted in *all* key field descriptions.

* ORDER NUMBER NOT WITHIN BOUNDS.
The order-1 number specified in the FIELD keyword parameter is less than 1 or greater than 255, or more than 255 key fields have been specified.

* RSOC AND FIELD INCONSISTENT, SPECIFIED TOGETHER.

* MAX-FILENUMBER, param, INCORRECT.
The max-filenumber parameter of the TAPES keyword parameter is less than 1 or greater than 14.

* TAPE, param LABEL-TYPE WRONG. STD USED.
The label-type parameter of the TAPES keyword parameter must be STD or NO.

* MORE THAN ONE MULTI-CYCLE PARAM. INCONSISTENT.

* RESERV WITH MULTI-CYCLE PARAMS INCONSISTENT.

* RESERV, param FILENAME INCORRECT.
Filename for the RESERV keyword parameter must be SM01, SM02, ..., SM14.

* SHARE WITH MULTI-CYCLE PARAMS INCONSISTENT.

* SHARE AND RESERV FILENAMES MUST BE UNIQUE.

* SHARE, param FILENAME INCORRECT.
Filename for the SHARE keyword parameter must be SM01, SM02, ..., SM14.

* DISC MAX-FILENUMBER, param, INCORRECT.
The max-filenumber value of the DISC keyword parameter is less than 1 or greater than 8.

* AUTO, param OUTPUT FILENAME INCORRECT.
Filename for the AUTO keyword parameter must be SM01, SM02,..., SM14.

* AUTO PARAMETERS MISSING.
The first two AUTO parameters are required.

* DOA PARAMETERS MISSING
The first two DOA parameters are required.

* DOA, param OUTPUT FILENAME INCORRECT.
Filename for the DOA keyword parameter must be SM01, SM02,..., SM14.

* DOA MERGE TAPES, param WRONG
The number of merge tapes specified in the DOA keyword parameter is less than 3 or greater than 14.

* DOM, param OUTPUT FILENAME INCORRECT.
Filename for the DOM keyword parameter must be SM01, SM02,..., SM14.

* CYCLE NO., param WRONG
The output cycle number of the DOM keyword parameter is less than 1.

* TOO MANY OR NOT-ENOUGH INPUT FILES.
The number of input files to merge, for the DOM keyword parameter, is less than 2 or greater than 13.

* IN-LABEL, param WRONG FORMAT
The in-label format of the DOM or DOF keyword parameter is uu/ccc where uu is a 2-character user identification and ccc is a three-character cycle number.

* TOO MANY IN-LABELS SPECIFIED.
The number of in-labels specified in the DOF keyword parameter is greater than 14.

* RESUME PASS PARAMETERS MISSING.

* RESUME CYCLE PARAMETERS MISSING.

* CYCLE NUMBER, param INCORRECT.
The RESUME=CYCLE or REDO=MERGE keyword parameter number is less than 1.

* RESUME MERGE PARAMETERS MISSING.

* IN-LABEL OR REEL NUMBER WRONG.
The RESUME=MERGE or REDO=MERGE keyword parameter format of in-label is incorrect or reel number is less than 1.

* RESUME FINAL PARAMETERS MISSING.

* RESUME KEYWORD, param INCORRECT.
The RESUME keyword parameter must be PASS, CYCLE, MERGE, or FINAL.

* REDO CYCLE PARAMETERS MISSING.

* CYCLE NUMBER, param INCORRECT.
Cycle number for the REDO=CYCLE keyword parameter is less than 1.

* REDO MERGE PARAMETERS MISSING.

* REDO KEYWORD, param INCORRECT.
The REDO keyword parameter must be CYCLE or MERGE.

* NOCKSM DEVICE, param INCORRECT.
The NOCKSM keyword parameter device must be T or D.

* CSPRAM, param, OPTION INCORRECT.
The CSPRAM keyword parameter option must be OPTION, YES, or NO.

APPENDIX F. SORT PARAMETER TABLE FORMAT

The sort parameter table is supplied to the Sort subroutine by means of the MR$OPN linkage. The MR$PRM macro instruction is provided to generate parameter table entries by means of keyword specifications. However, the table may be prepared by any other means. The table must be located on a word boundary. The first byte of each entry is a code value which identifies the entry. A hexadecimal FF code indicates a continuation of the entry.

CODE-CONTENT	KEYWORD PARAMETER	DESCRIPTION
00 aa aa aa		A word of binary zeros indicates the end of the parameter table. If the table is continued at another location, aaaaaa specifies the address of the next table (ADTABL parameter address).
01 aa aa aa	IN	aaaaaa – specifies the IN parameter address.
02 aa aa aa	OUT	aaaaaa – specifies the OUT parameter address.
03 aa aa aa	FIN	aaaaaa – specifies the FIN parameter address.
04 aa aa aa	RSOC	aaaaaa – specifies the RSOC parameter address.
05 aa aa aa	DROC	aaaaaa – specifies the DROC parameter address.
06 aa aa aa	DOEXT	aaaaaa – specifies the DOEXT parameter address.
07 aa aa aa FF nn nn nn	STOR	aaaaaa – specifies the STOR parameter address. nnnnnn – binary value of the number-of-bytes option. Specifies zero if number-of-bytes option is absent.
08 00 nn nn	RCSZ	nnnn – binary value of RCSZ maximum-byte (record size).
09 00 00 00	not used	
0A 00 nn nn	BIN (format 1, size/volume ratios absent)	nnnn – binary value of BIN bytes parameter.

CODE-CONTENT	KEYWORD PARAMETER	DESCRIPTION
0A 00 nn nn FF ss ss ss FF 00 vv vv . . . FF ss ss ss FF 00 vv vv	BIN (format 2, size/volume ratios present)	nnnn – binary value of BIN min-bytes parameter. ssssss – packed decimal value of BIN size-1 parameter. vvvv – packed decimal value of BIN vol-1 parameter. Each size-1, vol-1 parameter pair requires two BIN continuation words.
0B ll pp pp FF ff qq oo	FIELD	ll – binary value of FIELD length parameter, specified as 1 through 256, represented as $0 \leq ll \leq 255$. pppp – binary value of FIELD byte-position parameter, specified and represented as $0 \leq pppp \leq 16383$. ff – binary code of FIELD form parameter Format Code / Code (ff) CH (character) 00 BI (binary) 01 FI (fixed point integer) 02 PD (packed decimal) 03 ZD (zoned decimal) 04 qq – binary code of FIELD sequence parameter. 00 – ascending sequence, A 01 – descending sequence, D oo – binary value of FIELD order parameter $01 \leq oo \leq 255$.
0C 00 00 nn	DISC	nn – binary value specifying maximum number of disc filenames which may be assigned, $0 < nn \leq 8$.
0D 00 nn xx	TAPES	nn – binary value specifying maximum number of tape filenames which may be assigned, $0 < nn \leq 14$. xx – binary code indicating the TAPES label parameter. 00 – standard labels (STD) 01 – no labels (NO)

CODE-CONTENT	KEYWORD PARAMETER	DESCRIPTION
0E 00 00 bb	SHARE	bb – two unsigned decimal digits representing the value of the last two characters of the SHARE sort filename, that is, SM10, bb=10.
0F 00 00 bb	RESERV	bb – two unsigned decimal digits representing the value of the last two characters of the RESERV sort filename.
10 cc cc bb FF dd dd dd FF dd dd dd	AUTO	cccc – two-character output label-code parameter. bb – two unsigned decimal digits representing the value of the last two characters of the output sort filename parameter. dddddd – packed decimal value of the record-per-cycle parameter, decimal zero if the parameter is absent.
11 cc cc bb FF dd dd dd FF dd dd dd FF 00 00 tt	DOA	cccc – same as for AUTO bb – same as for AUTO dddddd – same as for AUTO tt – binary value of DOA number-merge-tapes parameter, zero if parameter is absent.
12 cc cc bb FF 00 yy yy FF ii ii ii FF ii ii ii . . . FF ii ii ii FF ii ii ii	DOM	cccc – same as AUTO bb – same as AUTO yyyy – packed decimal number of the DOM output cycle number. iiiiii – six-character in-label parameter. Two continuation words required for each in-label specified. Up to 13 in-labels are processed.
13 ii ii ii FF ii ii ii . . . FF ii ii ii FF ii ii ii	DOF	iiiiii – six-character in-label parameter. Two words required for each in-label parameter specified, first in-label is followed by two word continuation entries for each in-label. Up to 14 in-labels are processed.

CODE-CONTENT	KEYWORD PARAMETER	DESCRIPTION
14 rr rr rr	RESUME=(PASS,...)	rrrrrr – three-character PASS recovery-number parameter.
15 00 ee ee FF kk kk kk FF kk kk kk	RESUME=(CYCLE,...)	eeee – packed decimal value of cycle-number parameter. kkkkkk – packed decimal value of from-record-number parameter.
16 ii ii ii FF ii ii ii FF 00 yy yy	RESUME=(MERGE,...)	iiiiii – six-character tape-label parameter. yyyy – packed decimal value of reel-number parameter.
17 aa aa aa	RESUME=(FINAL,...)	aaaaaa – address of the RESUME=FINAL recovery information block.
18 00 ee ee FF kk kk kk FF kk kk kk FF uu uu uu FF uu uu uu	REDO=(CYCLE,...)	eeee – packed decimal value of cycle-number parameter. kkkkkk – packed decimal value of from-record-number parameter. uuuuuu – packed decimal value of to-record-number parameter.
19 ii ii ii FF ii ii ii FF 00 yy yy FF uu uu uu FF uu uu uu	REDO=(MERGE,...)	iiiiii – six-character tape-label parameter. yyyy – packed decimal value of reel-number parameter. uuuuuu – packed decimal value of the to-record-number parameter.
1A 00 gg hh	NOCKSM	gg-hh – binary code indicating checksum to be omitted. gg = 01 – omit tape checksum hh = 01 – omit disc checksum
20 00 00 00	PAD	Null pad entry used to reserve space in the parameter table. Entry is repeated (bytes+3)/4 times, where bytes is the PAD bytes parameter.
21 00 00 jj	CSPRAM	jj – binary code indicating CSPRAM option. 01 – OPTION 02 – YES 03 – NO


UNIVAC
SPERRY RAND