File No. S360/S370-33
Order No. GY28-6597-4

Program Logic

OS Sort/Merge Logic

Program Number 360S-SM-023

OS Release 21

This publication describes the internal logic of the OS sort/merge program. This program logic manual is directed to the IBM customer engineer who is responsible for program maintenance. It can be used to locate specific areas of the program, and it enables the reader to relate these areas to the corresponding program listings.

This version of the sort/merge program is designed to:

- Sort a data set using as intermediate storage the IBM 2400 Series (7- or 9-Tracks) Magnetic Tape Unit, or the IBM 3400 Series Magnetic Tape Units, or the IBM 2311 Disk Storage Drive, or the IBM 2314 Direct Access Storage Facility, or the IBM 2301 Drum Storage.

- Merge up to 16 previously sorted data sets.

Fifth Edition (January 1972)

This is a major revision of, and obsoletes,GY28-6597-3 and TNL GY33-8030. Descriptions of the following new features have been added.

Support for Advanced Checkpoint/Restart.
Support for Variable Spanned Records.
Support for 2420-7 Tape Drive.
Support for Blocked Input on SYSIN.

Changes to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to release 21 of IBM System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Order No.GN20-0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments appears at the back of this publication. If the form has been removed, comments may be addressed to the IBM Nordic Laboratory, Publications Development, Box 962, S-181 09, Lidingö 9, Sweden. Comments become the property of IBM.

Contents

Section 1: Introduction 7
Relationship to the Operating System . . 7
Structure of the Sort/Merge Program . . 7
Definition Phase 7
Optimization Phase 9
Sort Phase 9
Intermediate Merge Phase 10
Final Merge Phase 10
Initiating the Sort/Merge Program . . 10
Combining Sort/Merge Program Modules . 10

SECTION 2: SORT/MERGE PROGRAM THEORY OF OPERATION 17
Sorting Technique 17
Variable-Length Records 19
Fixed-Length Records 19
Sequence Distribution Techniques 20
Forcing Techniques 21
Intermediate Storage Requirements for Direct Access 22
2311, 2301, and 2314 with Balanced Technique 22
2314 with Crisscross Technique . . . 23
Balanced Direct Access Technique . . . 23
Crisscross Direct Access Technique . . 24
Balanced Tape Technique 28
Polyphase Tape Technique 29
Example of Sequence Distribution in Polyphase Tape Technique 29
Tables Used to Compute Sequence-Level Numbers 31
Oscillating Tape Technique 33
Merging Technique 36
Intermediate Merge Phase 36
Final Merge Phase 37
Record Movement in the Sort/Merge Program 37
Modules Used for Record Movement 37
Record Movement Techniques 39
Fixed-Length Records 39
Variable-Length Records 40

SECTION 3: PROGRAM ORGANIZATION 43
Sort System Interface 43
Definition Phase 43
Optimization Phase 43
Sort Phase 44
Selecting Modules for the Sort Phase . 44
Intermediate Merge Phase 45
Selecting Modules for the Intermediate Merge Phase 45
Final Merge Phase 46
Selecting Modules for the Final Merge Phase 46

SORT/MERGE PROGRAM FLOWCHARTS AND TABLES 48

SECTION 4: MODULE DIRECTORY 67

SECTION 5: DETAILED LAYOUTS 73
Overlay Structure 73
Storage Layouts 79
Phase-to-Phase Information (PPI) Area . 83

APPENDIX A: USER PROGRAM-MODIFICATION EXITS 95

APPENDIX B: REGISTER USAGE 96

APPENDIX C: MESSAGES PRODUCED BY THE SORT/MERGE PROGRAM 97

APPENDIX D: FORMAT CODES 101
Format Codes for Fixed-Length Records .101
Format Codes for Variable-Length Records 101
Condition Codes for Fixed- and Variable-Length Records 102

APPENDIX E: CHECKPOINT/RESTART FACILITY 103

APPENDIX F: PROGRAM LISTING STANDARDS AND CONVENTIONS 104
Module Names 104
Module Classifications 104
Instruction Names 104
Constant Names 104
Work Area Names 104
Table Names 104
Phase-to-Phase Information Area Names 105
Use of Routines in More Than One Module 105

Index 106

Illustrations

Figures

Figure 1. Sort Parameter List 8
Figure 2. Sort Initiation via LINK Macro 9
Figure 3. Combining Sort/Merge Program Modules 11
Figure 4. Replacement-Selection Technique 18
Figure 5. Tree Structure for Variable-Length Records With RSA of 17 - Records (16+1) 19
Figure 6. Tree Structure for Fixed-Length Records With RSA of 65 Records (64+1) 20
Figure 7. Example of Balanced Direct Access Sequence Distribution Technique 23
Figure 8. Example of Crisscross Direct Access Distribution Technique (Part 1 of 3) 25
Figure 9. Example of Balanced Tape Sequence Distribution Technique 29
Figure 10. Example of Polyphase Tape Sequence Distribution Technique 31
Figure 11. Example of Oscillating Tape Sequence Distribution Technique (Part 1 of 2) 34
Figure 12. Merging Technique Using Binary- Compare Network 36
Figure 13. Movement of Records in the Sort/Merge Program 37
Figure 14. Fixed-Length Record Format in the Record Storage Area 39
Figure 15. First Bin Format for Variable-Length Record in Record Storage Area 40
Figure 16. Continuation Bin Format for Variable-Length Record in Record Storage Area 41
Figure 17. Example of a Variable-Length Record in Contiguous Bins in Record Storage Area 41
Figure 18. Variable-Length Record Not in Contiguous Bins of Record Storage Area 42

Tables

Table 1. Sequence Distribution Technique Requirements 22

Charts

Chart 00. Structure of Sort/Merge Program 12
Chart 01. Sorting Applications Initiated by EXEC 13
Chart 02. Sorting Applications Initiated by ATTACH, LINK, or XCTL . . . 14
Chart 03. Merging Applications 15
Chart 04. Sorting Applications Using Linkage Editor 16
Chart 10. Overall Control of Flow in the Sort/Merge Program 49
Chart 20. Overall Organization Definition Phase 50
Chart 30. Overall Organization Optimization Phase 51
Chart 40. Overall Organization of Sort Phase for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques 52
Chart 41. Sort Phase Assignment for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques . . . 53
Chart 42. Sort Phase Decision Tables . 54
Chart 43. Sort Phase Decision Tables . 55
Chart 44. Sort Phase Decision Tables . 56
Chart 50. Overall Organization of Intermediate Merge Phase for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques 57
Chart 51. Intermediate Merge Phase Assignment for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques 58
Chart 52. Intermediate Merge Phase Decision Tables 59
Chart 53. Intermediate Merge Phase Decision Tables 60
Chart 60. Overall Organization of Sort/Merge Phases for Oscillating Tape and Crisscross Direct Access Techniques 61
Chart 61. Sort/Merge Phase Assignment for Oscillating Tape and Crisscross Direct Access Techniques 62
Chart 70. Overall Organization of Final Merge Phase 63
Chart 71. Final Merge Phase Assignment 64
Chart 72. Final Merge Phase Decision Tables 65
Chart 73. Final Merge Phase Decision Tables 66

Section 1: Introduction

The IBM System/360 Operating System sort/merge program is a generalized program that can sort and merge blocked and unblocked fixed-length or variable-length records based on control information supplied by the user. The formats and uses of the various sort/merge control statements are described in the publication OS Sort/Merge.

This version of the program is designed to:

- Sort a data set using one of the following devices for intermediate storage:

  IBM 3400 Series Magnetic Tape
  IBM 2400 Series Magnetic Tape
  IBM 2311 Disk Storage
  IBM 2301 Drum Storage
  IBM 2314 Direct Access Storage Facility

- Merge up to 16 previously sorted data sets.

Relationship to the Operating System

The sort/merge program is a processing program of the operating system and, as such, it communicates through macro instructions and interruptions with the following parts of the operating system control program:

- Job management routines that analyze job control statements and print messages.

- Task management routines that allocate main storage to various segments of the program and analyze conditions that have caused an interruption in the program.

- Data management routines that read data from and write data onto input/output devices.

The initial sorting and final merging I/O operations are performed by the data management QSAM routines. The EXCP macro instruction is used for intermediate storage I/O operations.

Structure of the Sort/Merge Program

The sort/merge program consists of five phases as shown in Chart 00. (Charts 00-04 are at the end of Section 1.)

- A definition phase, which reads and interprets sort/merge control statements and determines the sequence distribution technique to be used.

- An optimization phase, which optimizes the usage of channels and I/O devices.

- A sort phase, which arranges input records into ordered sequences and places the sequences onto intermediate storage devices.

- An intermediate merge phase, which merges the sequences produced by the sort phase into a lesser number of sequences and places them onto intermediate storage devices.

- A final merge phase, which, for a sort, combines the sequences produced by the intermediate merge phase into one sequence or, for a merge, combines the input data sets into one sequence.

DEFINITION PHASE

This phase is obtained from the link library (SYS1.LINKLIB) and selects the user routines, if any, to be link edited. It reads and interprets sort/merge control statements and selects one of five sequence distribution techniques to be used by the sort phase:

1. Balanced Direct Access, which is used when the intermediate storage device is a 2314 with six or fewer work areas, a 2301, or a 2311.

2. Crisscross Direct Access, which is used when a 2314 with more than six[1] work areas is the intermediate storage device.

3. Balanced Tape, which is always used whenever an estimated or exact input data set size is not given by the user

----

[1]Crisscross can be used on a 2314 when exactly six work areas are provided by the user if the user "forces" the technique, see "Forcing a Technique" in Section 2.

and if at least four tape intermediate storage data sets are provided by the user.

4. Polyphase Tape, which is always used whenever only three intermediate tape storage data sets are provided by the user.

5. Oscillating Tape, which may be used whenever the user supplies an exact or estimated data set size, provides more than three intermediate storage data sets, and does not specify the tape drive containing SORTIN as an intermediate storage device.

For sorting applications initiated by an EXEC statement, and for merging applications, the sort/merge control statements are provided in the SYSIN library. If user-modification routines are included, they are provided in the SYSIN data set and/or partitioned data sets. Any user routines appearing in SYSIN are copied onto a partitioned data set (SORTMODS).

For sorting applications initiated by an ATTACH, LINK, or XCTL macro instruction, the sort/merge control statements, user-modification routines and EXEC statement PARM field options, if any, are provided in main storage. The addresses of the control statements and user routines are passed to the sort/merge program by means of a parameter list. In addition to the addresses, this parameter list may contain some optional information. The format of the parameter list is illustrated in Figure 1.

The byte count defines the length (in bytes excluding itself) of the parameter list. It may be set to one of the following hexadecimal values, depending upon the number of optional entries included in the parameter list:

X'0018' -- if <u>none</u> of the optional entries is included.
X'001C' -- if <u>one</u> of the optional entries is included.
X'0020' -- if <u>two</u> of the optional entries are included.
X'0024' -- if <u>three</u> of the optional entries are included.
X'0028' -- if <u>four</u> of the optional entries are included.
X'002C' -- if all <u>five</u> of the optional entries are included.

All other values for this field are invalid.

<------------------------------4 bytes------------------------------>

Unused		Byte Count
Starting address of SORT statement[1]		
Ending address of SORT statement[1]		
Starting address of RECORD statement[1]		
Ending address of RECORD statement[1]		
Address of routine for exit E15 or zeros[1]		
Address of routine for exit E35 or zeros[1]		
Four alphameric characters[1], not BALN, OSCL, POLY, CRCX, or DIAG (to replace characters "SORT" in ddnames)		
X'00'	Optional main storage value[1]	
The characters DIAG (for diagnostic messages)		
X'FF'	Unused	Message printing control characters[1]
Sequence distribution technique selection characters[1] (BALN, OSCL, POLY, or CRCX)		

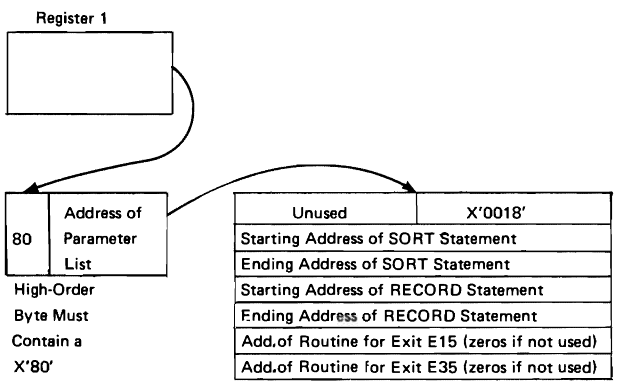
The last five entries (from "Four alphameric characters" through "Sequence distribution technique selection characters"): These entries are optional and, if included, may appear in any order.

[1]Refer to the publication <u>IBM System/360 Operating System: Sort/Merge</u> for a discussion of this field.

Figure 1. Sort Parameter List

The inclusion in the parameter list of an entry containing the characters DIAG causes the sort/merge program to print diagnostic messages, control statements, and a module map. Such an entry should be included only if a problem is encountered while trying to execute the sort/merge program. This entry must not be included in a normal sort environment because it will impair sort performance.

Figure 2 is an example of the use of the LINK macro instruction to initiate a sorting application. This figure also shows the format of the associated parameter list and illustrates the method by which the address of the parameter list is made available to the sort/merge program.



MACRO Example:

```
                LINK EP=SORT,PARAM=(LIST)
                •
                •
                •
                DS      0F
                DC      H'0'
LIST            DC      X'0018'
                DC      A(STRTSORT)
                DC      A(ENDSORT)
                DC      A(STARTREC)
                DC      A(ENDREC)
                DC      A(E15)
                DC      A(E35)
```

Figure 2. Sort Initiation via LINK Macro

The linkage editor will be called during the definition phase if the user has included modification routines which require link editing. (The user codes either a MODS statement with no fourth parameter or with S as a fourth parameter.) Refer to Chart 04 at the end of this section.

The linkage editor is obtained from the link library (SYS1.LINKLIB); it edits and combines the user routines and the associated phase modules.

Input to the linkage editor is a sequential data set (SYSLIN) containing INCLUDE statements which point to members in one of the following two locations:

- SORTMODS, which contains user-modification routines copied from SYSIN.
- User-specified data sets, which contain user-modification routines.

In addition to the above-mentioned data sets, the linkage editor uses SYSUT1 as a work device. When user routines request link editing, the output from the linkage editor is placed on SYSLMOD. The diagnostic output is placed on SYSPRINT, unless the sort/merge cataloged procedures, SORT or SORTD, are used, in which case no diagnostic output is produced.

The following names are given to the phases on SYSLMOD:

Sort Phase	S11,PH1	(For sorting applications only)
Intermediate Merge Phase	S21,PH2	(For sorting applications only)
Final Merge Phase	S31,PH3	(For sorting and merging applications)

OPTIMIZATION PHASE

The optimization phase is obtained from the link library (SYS1.LINKLIB) and optimizes the usage of the channel configuration of intermediate input/output devices.

SORT PHASE

The sort phase is obtained from the SORTLIB library. If any user routines are included in the program, they are obtained either from the SYSLMOD library created by the linkage editor or from the user-prepared library. This phase arranges the input records into ordered sequences according to the information in the user-specified control fields. It then writes these sequences onto intermediate storage devices (SORTWK01,...,SORTWK32) according to a predetermined sequence distribution procedure. Refer to Section 2 for a discussion of the "Sorting Technique" and "Sequence Distribution Techniques" implemented in the sort/merge program. The movement of records in the sort phase is described in Section 2 under "Record Movement in the Sort/Merge Program."

INTERMEDIATE MERGE PHASE

This phase is obtained from the SORTLIB library. If any user routines are to be used in this phase of the program, they are obtained either from the SYSLMOD library created by the linkage editor or from the user-prepared library. This phase combines (merges) the short sequences produced by the sort phase into a lesser number of longer sequences. These longer sequences are again written onto intermediate storage devices (SORTWK01,...,SORTWK32) according to a predetermined sequence distribution procedure. This phase is executed as many times as necessary until the number of sequences is less than or equal to the merge order. The sequence-combination method and the sequence-distribution procedures are described in Section 2 under "Merging Technique" and "Sequence Distribution Techniques," respectively. The movement of records in the intermediate merge phase is described in Section 2 under "Record Movement in the Sort/Merge Program."

In both the oscillating sort and crisscross sort, two of the five possible sequence distribution techniques, the sort and intermediate merge phases are kept in storage at the same time and control alternates between the two as directed by an algorithm. (Refer to the discussions of the Oscillating and Crisscross Techniques in Section 2.)

FINAL MERGE PHASE

The final merge phase is obtained from the SORTLIB library. Any user routines to be used in the phase are obtained either from the SYSLMOD library created by the linkage editor or from the user-prepared library. For a sorting application, this phase combines the remaining sequences on the intermediate storage devices (SORTWK01,..., SORTWK32) and produces a single sequence. For a merging application, this phase combines the sequences from the input devices (SORTIN01,...,SORTIN16) and produces a single sequence. This sequence is then written on the output device (SORTOUT).

The method used to combine the sequences is described in Section 2 under "Merging Technique." The movement of records in the final merge phase is described in Section 2 under "Record Movement in the Sort/Merge Program."

INITIATING THE SORT/MERGE PROGRAM

A sorting operation is initiated by either of two methods:

- An EXEC statement in the input stream. (See Chart 01.)
- An ATTACH, LINK, or XCTL macro instruction in another program. (See Chart 02.)

A merging operation is initiated by an EXEC statement in the input stream. (See Chart 03.)

COMBINING SORT/MERGE PROGRAM MODULES

The modules required for a specific sort or merge must be combined at program execution time. Figure 3 shows how the sort/merge program modules and user routines are combined.

During generation of the operating system, sort/merge program modules are copied from SYS1.SM023 and placed in the sort library, SYS1.SORTLIB. The modules needed for the definition phase and optimization phase are link edited into the link library, SYS1.LINKLIB. The modules for the sort, intermediate merge, and final merge phases are placed in the sort library.

When the sort/merge program is to be executed, the definition phase modules are obtained from SYS1.LINKLIB. Sort/merge control statements are obtained either from SYSIN or from main storage.

If the linkage editor is required, it is obtained from SYS1.LINKLIB as are the optimization phase modules. Modules for the sort phase are obtained from SYS1.SORTLIB. Modules for the intermediate merge phase and final merge phase come from SYS1. SORTLIB. User routines are picked up from SYSLMOD if they were link edited during the current sort/merge execution, or from a user library if they were link edited previously and did not require further link editing.

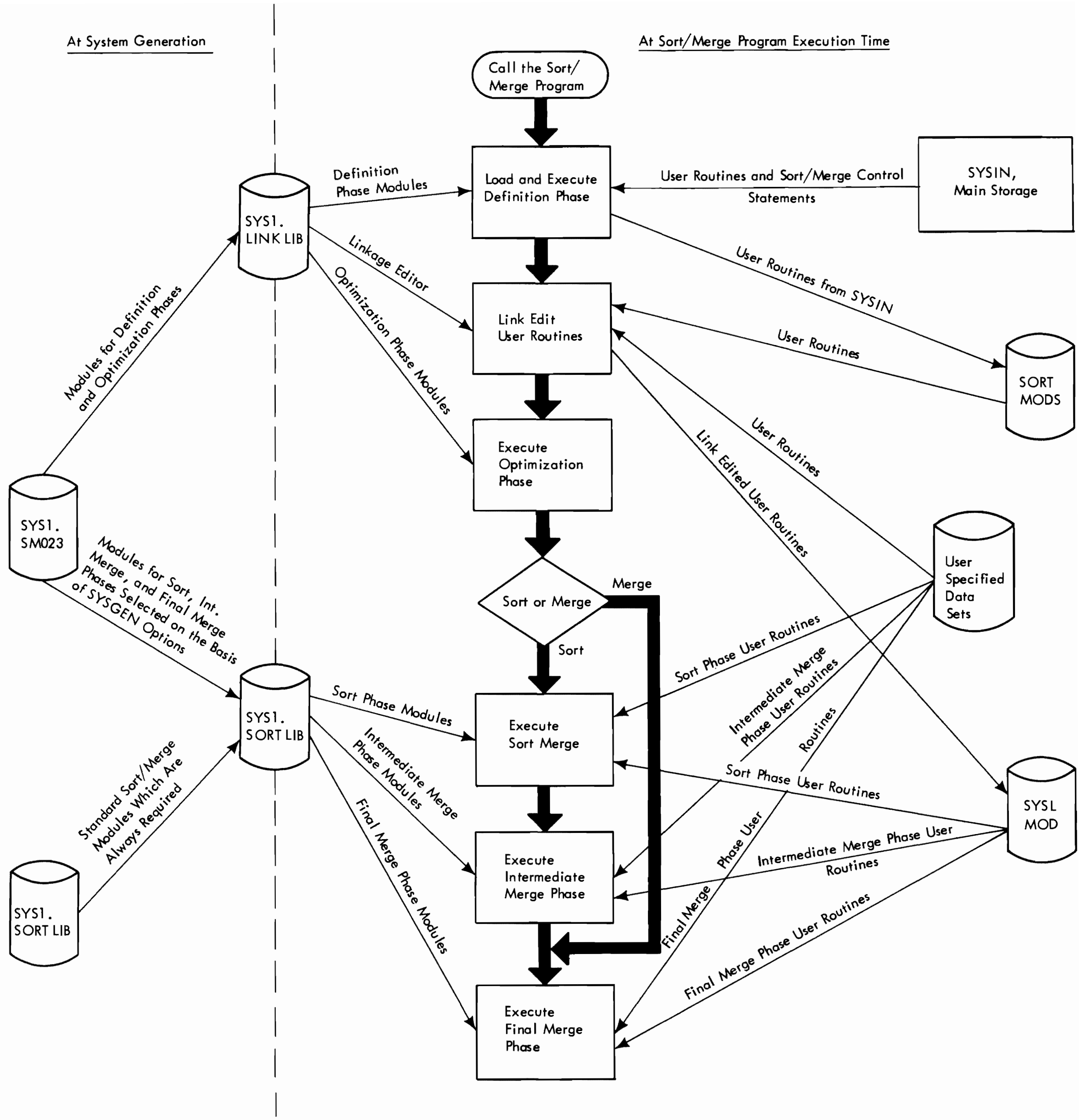


Figure 3. Combining Sort/Merge Program Modules

Chart 00. Structure of Sort/Merge Program

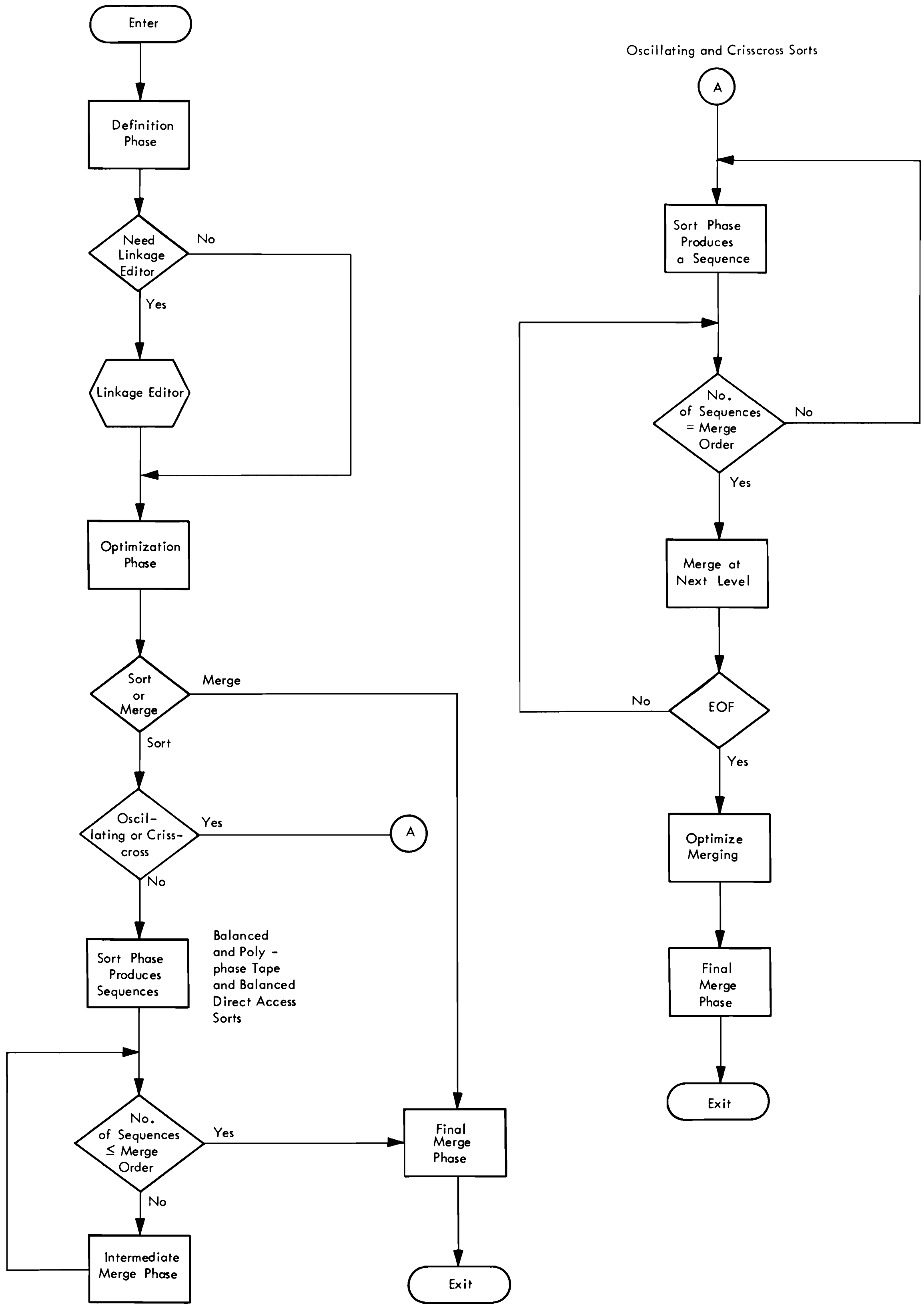


Chart 01. Sorting Applications Initiated by EXEC

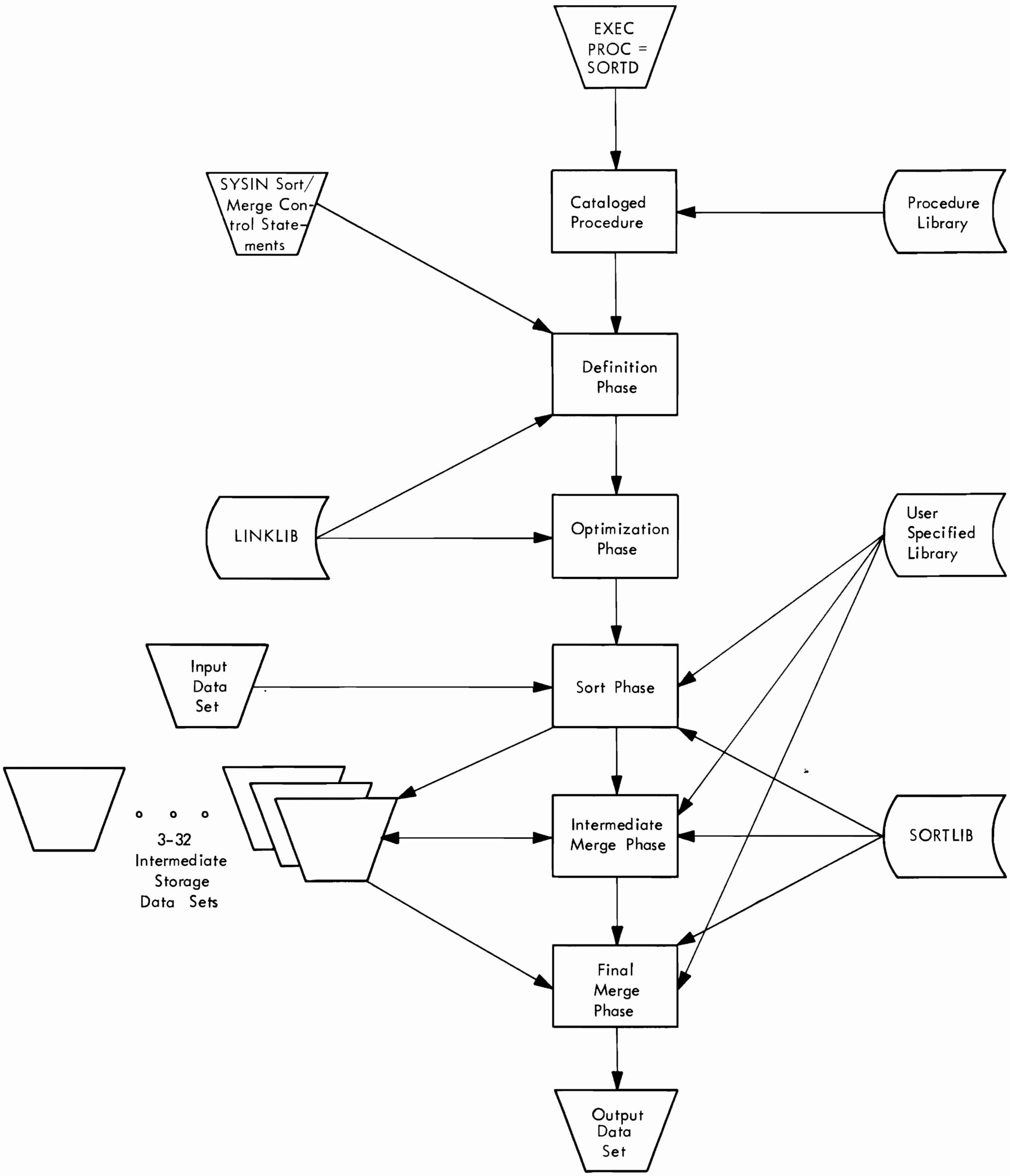


Chart 02. Sorting Applications Initiated by ATTACH, LINK, or XCTL

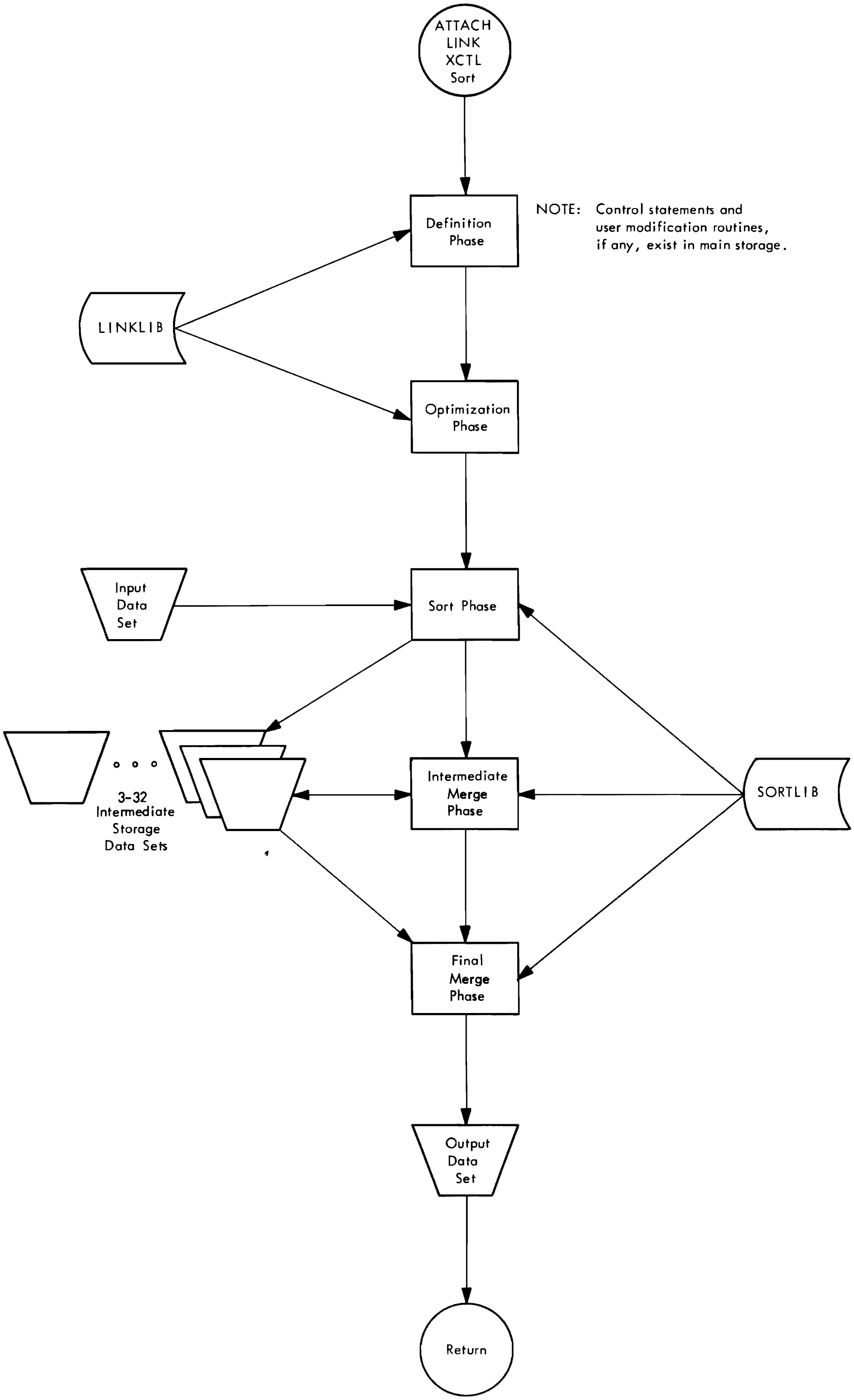


Chart 03. Merging Applications

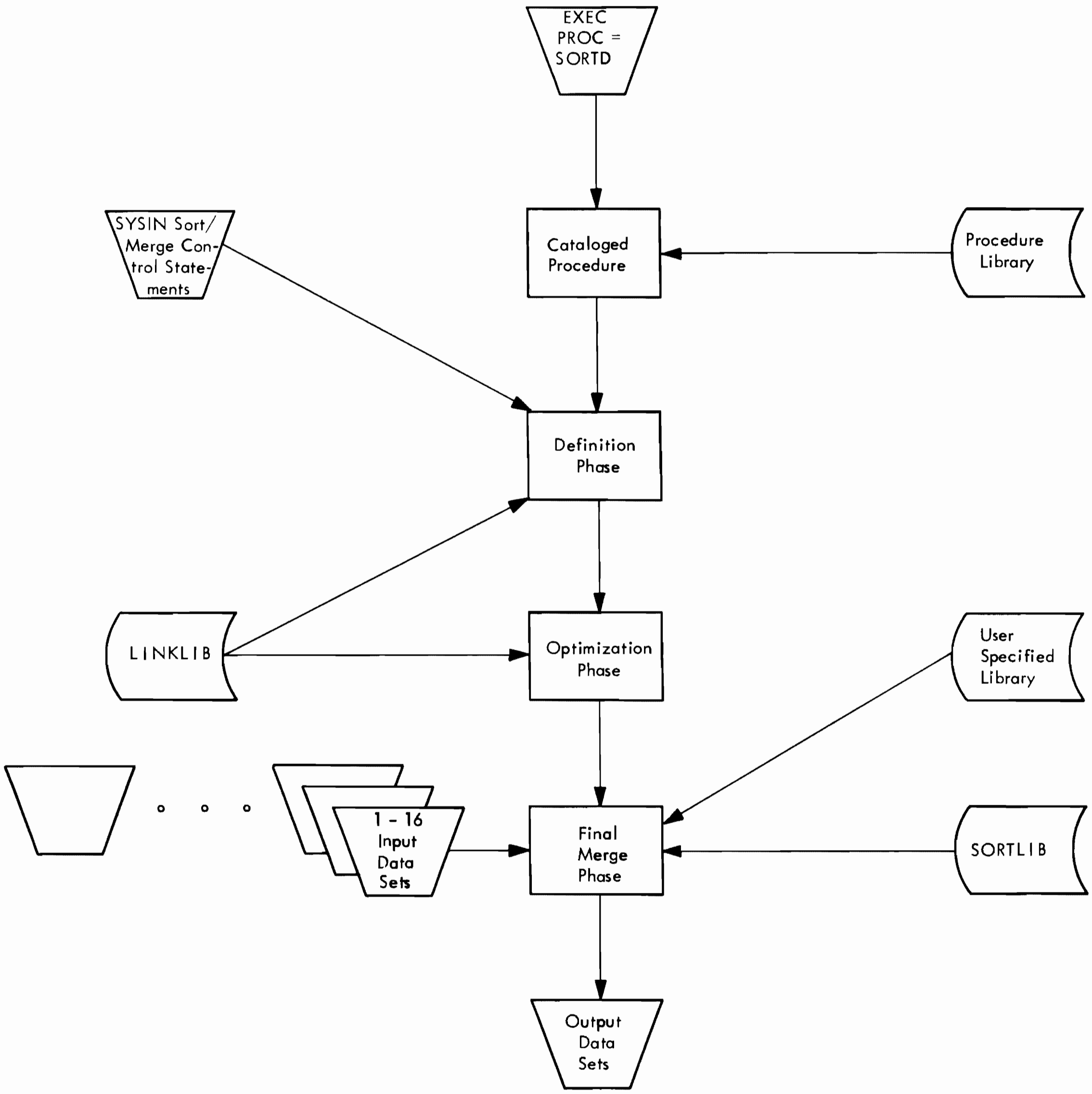
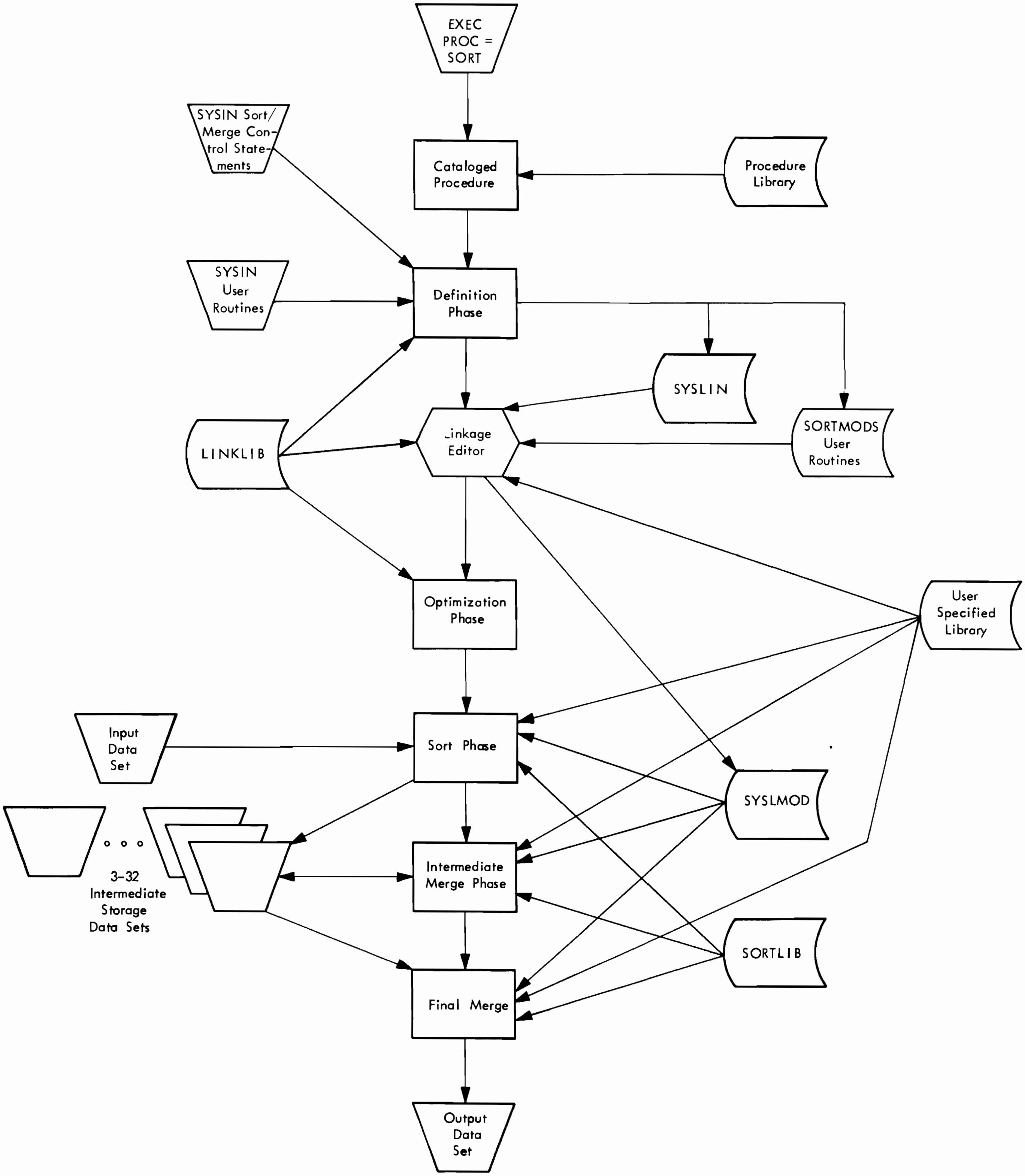


Chart 04. Sorting Applications Using Linkage Editor



Section 2: Sort/Merge Program Theory of Operation

This section describes the sorting and merging techniques used by the sort/merge program. The section also describes the format and movement of records from the time they are read in from the input device until they are placed on the output device.

Sorting Technique

The sort/merge program uses the "replacement-selection technique" to sort records. This technique:

1. Reads a block of records from the input data set to an input buffer.
2. Moves one of the records from the input buffer into the record storage area (RSA). Variable-length spanned records (VRE) are moved from the input buffer to a work area, which is large enough to contain the largest record in the input data set. There the segments of the spanned record are gathered before the record is moved to the RSA.
3. Determines which record in the RSA should be placed next in an ordered sequence.
4. Moves that record to an output buffer.
5. Replaces the moved record with the next record from the input buffer.
6. Repeats from 2 until the input buffer is empty, and then repeats from 1 until the input data set end-of-file.

If the new record should follow the selected output record in the current sequence, the new record forms a part of the group of records from which the next output record is selected. If the new record should precede the selected output record, the new record is retained in RSA for the next sequence. The sorting process continues for the current sequence until there are no more replacement records that fit in the current sequence. The last block of records in the current sequence is then written, a new sequence is started, and the procedure is repeated until the entire input data set has been processed.

The RSA has space for one more record than the number of records that can be sorted at one time. After a record is selected for output, it remains in the RSA and a new record is read into the extra space. After the new record and the output record have been compared to determine whether the new record fits in the current sequence, the output record is moved to an output buffer and the space it occupied in the RSA is made available for another new record.

The RSA is primed with a group of records from the input buffer. Sort/merge then selects the first output record from the contents of the RSA. In Figure 4, sort/merge selects record 5 as the first output record. Before placing record 5 in the output buffer, however, sort/merge brings the first record from the input buffer, record 75 into the RSA and compares it with the selected output record. Record 5 becomes the first output record. Record 6 is selected as the second output record. Record 91 is brought in from the input buffer and is compared with record 6. Record 6 becomes the second output record and record 91 takes the place of record 6 in the RSA. Record 11 is then selected for output and is compared with record 3 which has just been brought in from the input buffer. Record 3 has arrived too late for this sequence; record 11 becomes the third output record and record 3 is flagged to indicate that it must wait for the next sequence. Record 3 takes the place of record 11 in the RSA. This process continues until none of the records in the RSA fit in the current sequence. Sort/merge then concludes that sequence and begins a new one.

The replacement-selection technique puts records in sequence as they come into the RSA and keeps track of RSA addresses by forming a "tree structure." The tree structure may be defined as an area of main storage divided into "nodes," each containing information about the comparison of either two or four records in the RSA, depending on the network used. The nodes are grouped into "levels," each representing one of a series of record comparisons. (Refer to Appendix D for a description of how nodes are constructed.)

In either an oscillating or crisscross sort, on every Mth sequence, the sort phase proceeds as if an EOF had occurred, and the tree is flushed. (M is the merge order and is equal to N-1, where N is the number of intermediate work units.) Then control is passed to the intermediate merge phase. When merging is complete, the sort process is reinitialized; i.e., RSA is filled again and production of sequences continues for another M sequences or until an actual EOF has occurred.

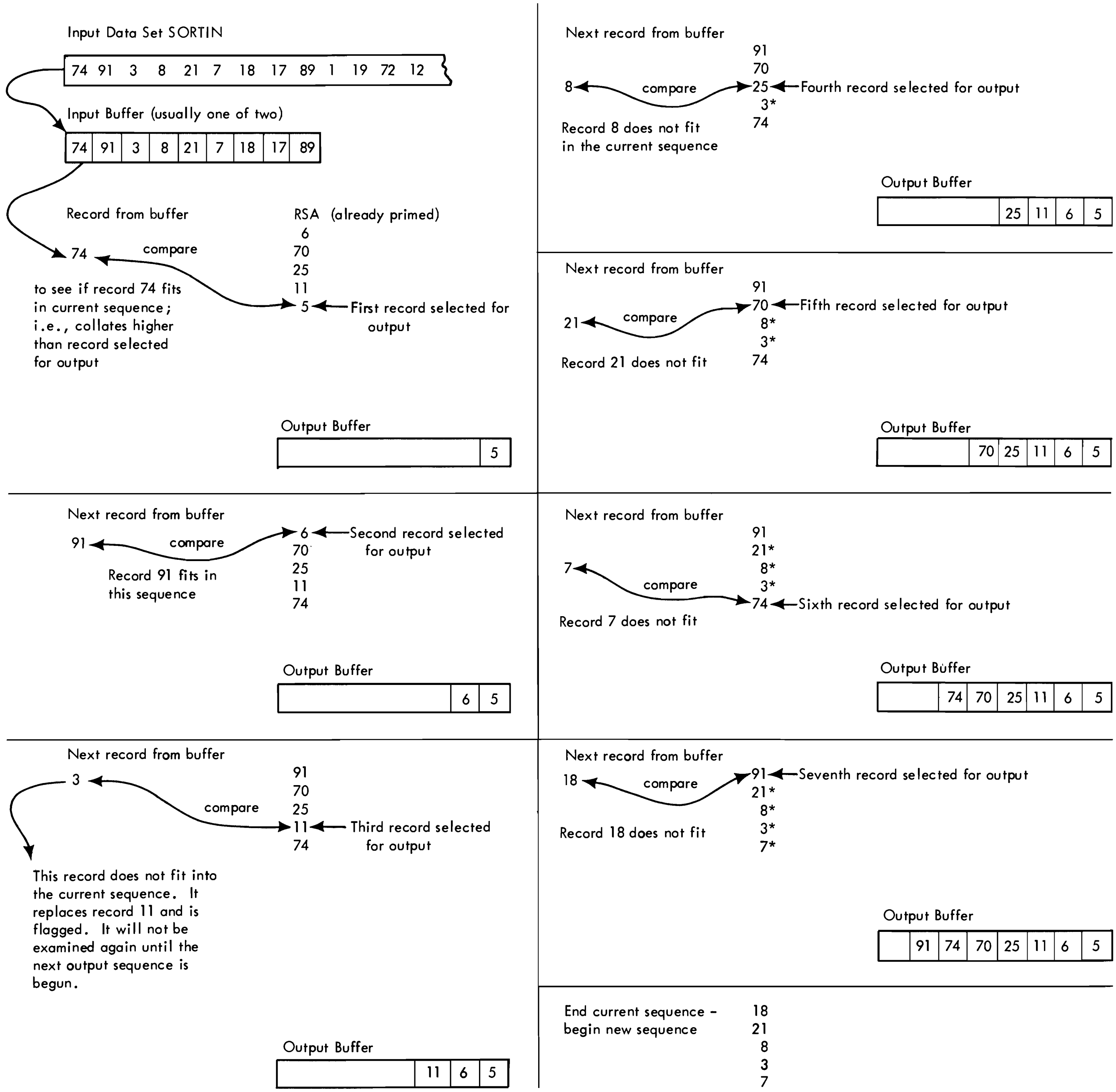


Figure 4. Replacement-Selection Technique

VARIABLE-LENGTH RECORDS

Records are brought into the RSA one at a time. Associated with each pair of records is the address of a first-level node. The records in each pair are compared to determine which record in the pair should appear first in an ordered record sequence (i.e., which record collates lower for ascending sequences or which collates higher for descending sequences). The record chosen from each pair is called the "winner" record; that not chosen is called the "loser" record.

After the comparisons are made at the first-level nodes, the addresses of the loser records are recorded in the nodes. The winner records are then compared at the second-level nodes referred to by the next level addresses in the first-level nodes. (See Figure 5.)

The comparisons at the second-level nodes produce loser records whose addresses are kept in the nodes, and winner records that are compared at the third-level nodes. The process continues until there is a single winner record selected from the comparison at the last-level node. The selected record is the first record to be written in an ordered sequence.

After a record is selected for output, a new record is brought into the empty space in the RSA. The new record is associated with the same first-level node as the record just selected for output.

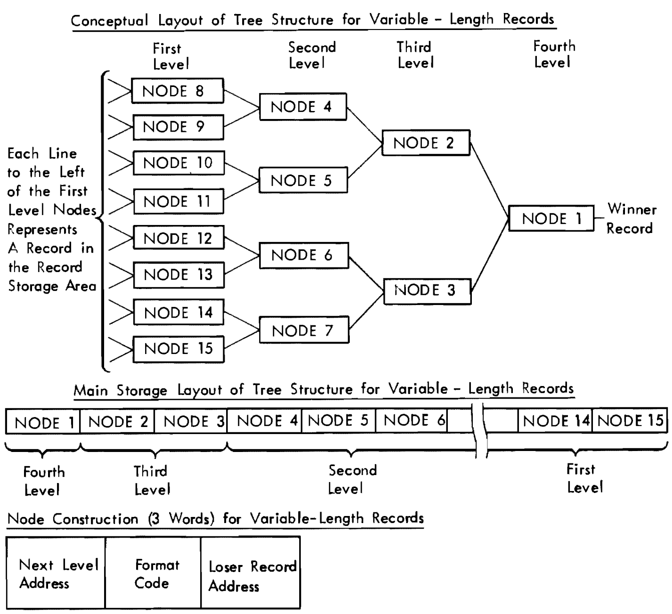


Figure 5. Tree Structure for Variable-Length Records With RSA of 17 Records (16+1)

The new record is compared to the output record. If the new record precedes the selected output record in the current sequence, it is retained for the next sequence. The new record's address is flagged and remains in the tree. After the new record and the output record are compared, the output record is moved to an output buffer. The space in the RSA occupied by the output record is now considered empty.

The new record, whether flagged or not, is then compared to the loser record referred to by the record address in the relevant first-level node. This comparison produces a loser record and a winner record. Flagged records always lose. The address of the loser record is kept in the node. The winner record is compared to the loser record whose address is in the second-level node referred to by the first-level node. This comparison produces a loser record and a winner record. Successive node comparisons continue until a winner record is selected for output. Another record is moved into the RSA and the above process is repeated until the entire input data set has been processed.

The node structure is established at assignment time and the format code indicates that it is empty. After the last record is introduced, the remaining records are flushed out.

FIXED-LENGTH RECORDS

The tree structure for 64 fixed-length records (RSA=65) is shown in Figure 6.

The sorting technique used for fixed-length records is similar to that used for variable-length records except for the following differences:

1. The address of each first-level node is associated with four records in the RSA.

2. At each level, the new record is ordered with three loser records at that level.

3. Each node contains information about the comparison of four records. Hence, records are ordered in groups of four.

4. After each grouping, there are three loser records and one winner record; each node has space for three loser record addresses.

5. The format code in a node indicates the status of the node's three loser records.

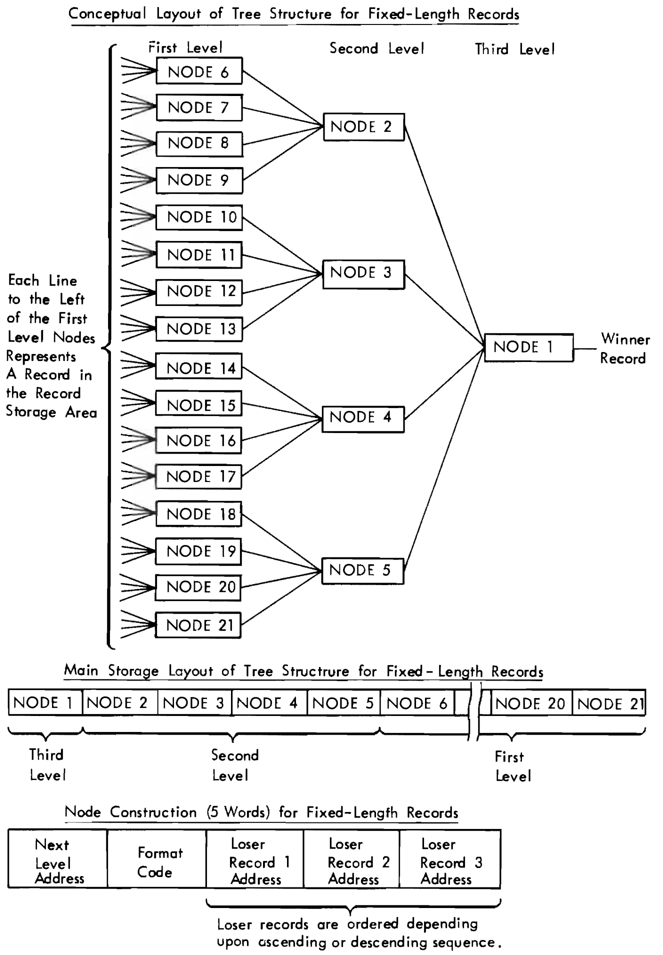


Figure 6. Tree Structure for Fixed-Length Records With RSA of 65 Records (64+1)

Sequence Distribution Techniques

The sort/merge program can use five different techniques for sequence distribution. Each technique differs in the way in which sequences are distributed onto the intermediate storage devices and in the order in which the number of intermediate merge passes are reduced. The five techniques are:

1. Balanced direct access technique.
2. Crisscross direct access technique.
3. Balanced tape technique.
4. Polyphase tape technique.
5. Oscillating tape technique.

The balanced direct access technique is used if the intermediate work areas are on 2301 drum storage or 2311 disk storage. If the intermediate work areas are on a 2314 direct access storage facility, either the balanced or crisscross direct access technique is used depending on the number of work areas available as follows:

1. If less than six work areas are provided by the user, the balanced technique is used.

2. If more than six work areas are provided, the crisscross technique is used.

3. If exactly six work areas are provided, the balanced technique is used unless the user forces the crisscross technique.

If the intermediate storage medium is tape, either the balanced, polyphase, or oscillating tape technique is used. The program evaluates the sort parameters specified by the user to determine which of the three possible tape techniques it will use. The program needs an exact or closely estimated input data set size (SORT statement SIZE operand) to select the most efficient tape technique. If the user does not specify a file size, sort/merge chooses the polyphase technique if only three intermediate storage tapes are available, or the balanced technique if four or more tapes are available.

Sort/merge evaluates the following sort parameters to determine which tape technique to use:

1. Input data set size -- exact, estimated, or omitted.

2. N -- the number of intermediate storage tapes available to the sort.

3. Tape densities.

4. Amount of main storage available to the sort.

5. Channel configuration (multiplex, 1 selector, 2 selectors, read-while-write tape control unit, or tape switch).

6. User input/output blocking factors.

Maximum input for a tape sort varies with technique. The formulas for tape technique capacities are as follows, where N is the number of intermediate storage tapes:

Oscillating tape technique -- maximum input is N-2 reels of tape at sort blocking.[1] $4 \le N \le 17$.

----

[1]The publication IBM System/360 Operating System: Sort/Merge Timing Estimates contains sort blocking factors for various combinations of record lengths and main storage values.

Balanced tape technique -- maximum input is (N/2)-1 reels of tape at sort blocking. $4 \leq N \leq 32$.

Polyphase tape technique -- maximum input is 1 reel of tape at sort blocking. $3 \leq N \leq 17$.

If only three intermediate storage tapes are available, sort/merge always chooses the polyphase technique. If the input unit is also specified as an intermediate storage unit, the oscillating technique cannot be used. The balanced and polyphase techniques require the input unit as a work unit only after the entire input file has been processed, whereas the oscillating technique requires the input unit as a work unit after N-1 input strings have been processed.

If the number of intermediate storage tapes available to the sort exceeds 17, the polyphase and oscillating techniques are evaluated using 17 units as maximum. The balanced technique is evaluated with up to 32 available units. The polyphase and oscillating techniques, with their higher merge orders, may allow a more efficient sort despite the reduction in the number of intermediate storage units used. This will also expand the capacity of the specified sort.

FORCING TECHNIQUES

The user can force a particular technique to be used for a sorting application. However, since the sort/merge program attempts to select the most efficient technique, the user should be aware that forcing a technique can seriously impair sort/merge performance.

Table 1 shows the requirements of the five techniques. If the user forces a technique, but does not provide sufficient main storage or intermediate storage, sort/merge will select another technique rather than terminate the sorting application.

The method used to override the sort/merge program and force a particular technique is governed by the manner in which the sort is initiated. If the sort is initiated by an EXEC statement, overriding is effected by including one of the following parameters in the PARM field of that statement:

- BALN -- for the balanced tape or balanced direct access technique
- OSCL -- for the oscillating tape technique
- POLY -- for the polyphase tape technique
- CRCX -- for the crisscross direct access technique

If the sort is initiated by an ATTACH, LINK, or XCTL macro instruction, overriding is effected by including one of the above parameters as an optional entry in the sort parameter list. (Refer to Figure 1.)

Table 1. Sequence Distribution Technique Requirements

Technique	Minimum Main Storage For Sort/ Merge	Maximum Input	Minimum Intermediate Storage Areas Required	Maximum Intermediate Storage Areas Permitted	Comments
Balanced Tape BALN	12,000 bytes	15 reels	2 (x+1), where x is the number of input volumes	32 tape units	Always used if more than three intermediate storage tapes are available and input data set size is not specified or estimated.
Polyphase Tape POLY	12,000 bytes	1 reel	3 reels	17 tape units	Always used if only three intermediate storage tapes are available.
Oscillating Tape OSCL	21,000 bytes	15 reels	x+2 or 4, whichever is greater, where x is the number of input volumes	17 tape units	Input data set size must be given or closely estimated. The tape drive containing SORTIN, cannot be assigned as an intermediate storage unit.
Balanced Direct Access BALN	13,000 bytes	No fixed maximum-depends on available main storage and available intermediate storage	3 areas	6 areas	The only technique available for the 2301 and 2311. Always used on 2314 when less than six work areas are available. Used on 2314 when six areas are available unless CRCX is forced.
Crisscross Direct Access CRCX	24,000 bytes		6 areas	17 areas	Always used on 2314 when more than six work areas are available. Used on 2314 when six areas are available but must be forced. Not used on 2301 or 2311.

INTERMEDIATE STORAGE REQUIREMENTS FOR DIRECT ACCESS

2311, 2301, and 2314 with Balanced Technique

Total number of tracks = $\frac{S(N)}{k(N-1)} + 2N$

where

N
: is the number of intermediate storage areas $3 \leq N \leq 6$

S
: is the number of records in the input data set, exact or approximate.

k
: is B/L

B
: is 3400 for the 2311
18000 for the 2301
7000 for the 2314

L
: is the length in bytes of the records in the input data set; maximum length for variable length records.

Only the integer portion of k is used. If the formula yields k=0, use k=1.

2314 with Crisscross Technique

Total number of tracks = $\frac{1.25S}{k}$

where

S
: is the number of records in the input data set, exact or approximate.

k
: is B/L

  B
  : is 7000

  L
  : is the length in bytes of the records in the input data set; maximum length for variable length records.
  Only the integer portion of k is used. If the formula yields k=0, use k=1.

BALANCED DIRECT ACCESS TECHNIQUE

The sort phase distributes sequences onto all but the largest area set aside for intermediate storage. The order (ascending or descending) of the control fields of all the sequences is the same as the order desired for output.

Each area has a directory in which the locations of individual sequences are maintained. The directory for each area resides in that area and is pointed to by a parameter in the phase-to-phase information area (PPI). For example, if an intermediate storage area contains 50 tracks, sort/merge begins writing sequences on the first track and places the directory on the last track. One track is always used for the directory and more tracks may be required depending on the number of sequences placed in the work area. An additional track immediately preceding the directory area is reserved for EOF processing.

The intermediate merge phase combines sequences from one of the filled areas into longer sequences and places these sequences onto the empty area. When the filled area has been completely exhausted, it is considered empty and can then receive sequences from some other area. The merging process continues until the total number of sequences in all the areas is less than or equal to the maximum possible merge order. At that time, the final merge phase combines the remaining sequences into a single sequence and places it onto the output device.

Figure 7 shows an example of the balanced direct access technique. Area A is the same size or smaller than area B, which is the same size or smaller than area C. Due to the amount of main storage available for this example, 5 is the maximum merge order that could be chosen. Merge orders of 4-4-3 are selected as most efficient for this example. The sort phase distributes 22 sequences onto area A and 16 sequences onto area B. The intermediate merge phase merges the 16 sequences from B into 4 longer sequences and places them on C. Since C is equal to or larger than B, all the sequences from B fit on C.

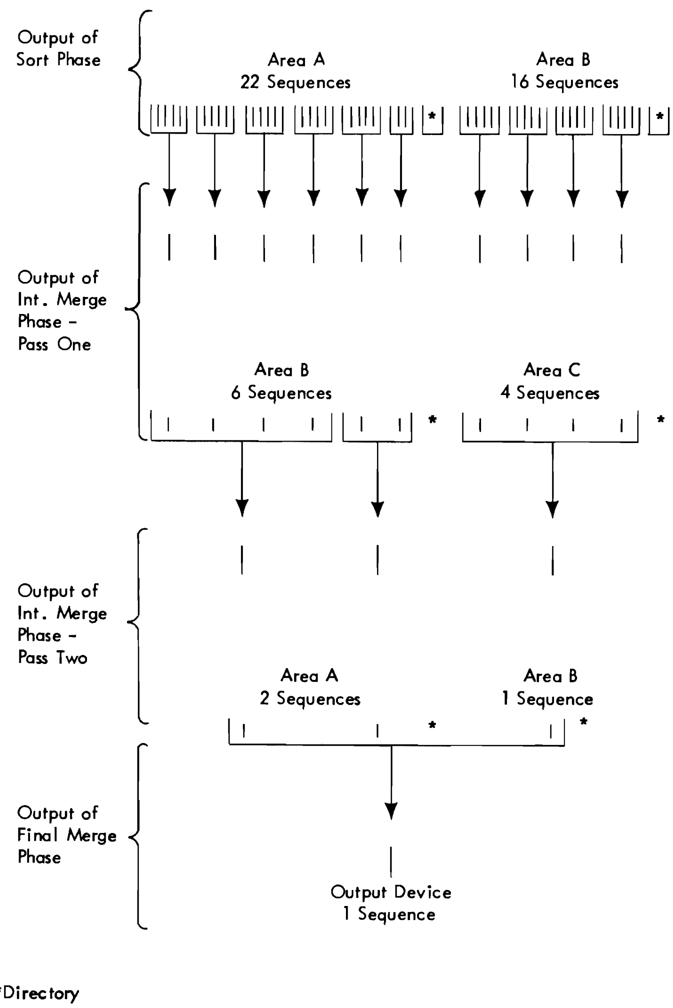


Figure 7. Example of Balanced Direct Access Sequence Distribution Technique

B is now considered empty and can receive sequences from A. The 22 sequences on A are merged into 6 longer sequences and placed on B. A is now empty. Since the total number of sequences (10) on B and C is greater than the maximum possible merge order (5), another intermediate merge phase pass is required. The 6 sequences from B are merged into 2 sequences and placed on A. Since B contains sequences originally from A, the sequences fit on A. B is now empty. The 4 sequences from C are merged into one long sequence, which is placed on B. The total number of sequences on A and B is now less than the maximum possible merge order, so the final merge phase combines the remaining 3 sequences into a single sequence and places it onto the output device.

CRISSCROSS DIRECT ACCESS TECHNIQUE

The merge order for the crisscross technique is one less than the number of intermediate storage areas available to sort/merge. Control alternates between the sort phase, which produces and distributes sequences, and the intermediate merge phase, which combines the sequences into longer sequences. The crisscross technique merges the shorter sequences first and delays handling the longer sequences until absolutely necessary. This action minimizes the amount of data handled during each intermediate merge phase pass and results in more efficient operation. Another advantage of the crisscross technique is its ability to sort large input data sets. Crisscross is used only on the 2314 and only when six or more intermediate storage areas are available.

The crisscross sort begins by distributing, via the sort phase, a sequence onto all but one of the intermediate storage areas. Then the intermediate merge phase combines the sequences into a longer sequence and places it on the empty area. The sort phase then creates and distributes sequences onto all but one of the areas (a different area this time) and the intermediate merge phase combines them and places the resulting sequences onto the area that did not receive a sequence. This continues until all but one of the areas contain a longer sequence formed from the original sequences created by the sort phase. This point in crisscross is termed the first base level. If the number of intermediate storage areas is referred to as N, base levels are attained whenever all but one of the areas contains (N-1) K sequences, where K is an integer greater than zero. For example, if the number of intermediate storage areas is six, the first base level occurs when all but one of the areas contains a sequence made up of five original sequences. The second base level occurs when all but one of the areas contains a single sequence formed from 25 original sequences. Whenever a base level occurs, one of the areas is empty.

Note: Each of the N-1 sequences that exists at the time a base level is attained is referred to as a base sequence.

After the first base level is attained, the crisscross sort distributes a sequence onto each of N-1 work areas and merges these sequences into a single sequence, which is placed onto the Nth work area. It does this N-2 times, varying the work areas onto which the N-1 sequences are distributed. At this point, N-2 additional sequences, each formed from N-1 original sequences, have been made. The crisscross sort then merges the N-2 additional sequences with one of the base sequences to form a sequence that is made up of $(N-1)^2$ original sequences. It does this entire process N-1 times to yield the second base level. The sequences at the second base level are each formed from $(N-1)^2$ original sequences. Continuing in this manner, the crisscross sort develops successive base levels whose base sequences are formed from $(N-1)^3$, $(N-1)^4$,... original sequences, until the end-of-file is reached. At this point, the crisscross sort carries out successive N-1 way merges to reduce the number of remaining sequences to N-1 or fewer. Control is then passed to the final merge phase to complete the sorting application.

Note: As the crisscross sort proceeds to higher base levels (e.g., from base level $(N-1)^2$ to base level $(N-1)^3$), successive merges, without an intervening distribution of sequences, may occur after each group of N-2 cycles (i.e., after N-1 sequences have been distributed and then merged into one longer sequence a total of N-2 times).

Figure 8 illustrates the crisscross sort technique using 125 sequences and six work areas named A, B, C, D, E, and F.

The method of selecting the work areas into which the N-1 sequences are distributed is described in the comments column of Figure 8.

Distribution Cycle Number	Work Areas and Sequence Arrangement						Operation Performed	Comments (if any)
	A	B	C	D	E	F		
1	1^1	1^2	1^3	1^4	1^5		Distribute N-1 sequences	This cycle distributes sequences onto each of N-1 work areas.
						5	Merge to Nth area	
2	1^2	1^3	1^4	1^5		5 1^1	Distribute N-1 sequences	Each of the next N-2 cycles distributes: (a)the first of the N-1 sequences onto the work area that did not receive a sequence during the previous cycle; (b)the second of the N-1 sequences onto the work area that received the first sequence during the previous cycle; (c) the third of the N-1 sequences onto the work area that received the second sequence during the previous cycle; etc.
					5	5	Merge to Nth area	
3	1^3	1^4	1^5		5 1^1	5 1^2	Distribute N-1 Sequences	
				5	5	5	Merge to Nth area	
4	1^4	1^5		5 1^1	5 1^2	5 1^3	Distribute N-1 sequences	
			5	5	5	5	merge to Nth area	
5	1^5		5 1^1	5 1^2	5 1^3	5 1^4	Distribute N-1 sequences	
		5	5	5	5	5	Merge to Nth area	First base level attained. (Processing to this point is identical to the oscillating tape sort, but, from here on, deviation occurs.)
6	1^2	5 1^1	5	5 1^5	5 1^4	5 1^3	Distribute N-1 sequences	This cycle distributes: (a) the first of the N-1 sequences onto the work area that did not receive a sequence during the previous cycle; (b) the second of the N-1 sequences onto the work area that received the last sequence during the previous cycle; (c) the third of the N-1 sequences onto the work area that received the next-to-the-last sequence during the previous cycle; etc. Thus, this cycle changes the direction of the distribution.
		5	5 5	5	5	5	Merge to Nth area	
7	1^3	5 1^2	5 5 1^1	5	5 1^5	5 1^4	Distribute N-1 sequences	Each of the next N-3 cycles distributes the N-1 sequences in the manner described for cycles 2 through 5 above.
		5	5 5	5 5	5	5	Merge to Nth area	
8	1^4	5 1^3	5 5 1^2	5 5 1^1	5	5 1^5	Distribute N-1 sequences	
		5	5 5	5 5	5 5	5	Merge to Nth area	
9	1^5	5 1^4	5 5 1^3	5 5 1^2	5 5 1^1	5	Distribute N-1 sequences	
		5	5 5	5 5	5 5	5 5	Merge to Nth area	
	25		5	5	5	5	Merge to produce a sequence made up of $(N-1)^2$ original sequences	

Figure 8. Example of Crisscross Direct Access Distribution Technique (Part 1 of 3)

Distribution Cycle Number	Work Areas and Sequence Arrangement						Operation Performed	Comments (if any)
10	25 1^3	 1^2	5 1^1	5	5 1^5	5 1^4	Distribute N-1 sequences	This cycle distributes the N-1 sequences in the same manner as they were distributed N-3 cycles prior to it (in cycle 7).
	25		5	5 5	5	5	Merge to Nth area	
11	25 1^4	 1^3	5 1^2	5 5 1^1	5	5 1^5	Distribute N-1 sequences	Each of the next N-3 cycles distributes the N-1 sequences in the manner described for cycles 2 through 5 above.
	25		5	5 5	5 5	5	Merge to Nth area	
12	25 1^5	 1^4	5 1^3	5 5 1^2	5 5 1^1	5	Distribute N-1 sequences	
	25		5	5 5	5 5	5 5	Merge to Nth area	
13	25	 1^5	5 1^4	5 5 1^3	5 5 1^2	5 5 1^1	Distribute N-1 sequences	
	25 5		5	5 5	5 5	5 5	Merge to Nth area	
	25	25		5	5	5	Merge to produce second sequence made up of $(N-1)^2$ original sequences	
14	25 1^4	25 1^3	 1^2	5 1^1	5	5 1^5	Distribute N-1 sequences	This cycle distributes the N-1 sequences in the same manner as they were distributed N-3 cycles prior to it (in cycle 11).
	25	25		5	5 5	5	Merge to Nth area	
15	25 1^5	25 1^4	 1^3	5 1^2	5 5 1^1	5	Distribute N-1 sequences	Each of the next N-3 cycles distributes the N-1 sequences in the manner described for cycles 2 through 5 above.
	25	25		5	5 5	5 5	Merge to Nth area	
16	25	25 1^5	 1^4	5 1^3	5 5 1^2	5 5 1^1	Distribute N-1 sequences	
	25 5	25		5	5 5	5 5	Merge to Nth area	
17	25 5 1^1	25	 1^5	5 1^4	5 5 1^3	5 5 1^2	Distribute N-1 sequences	
	25 5	25 5		5	5 5	5 5	Merge to Nth area	
	25	25	25		5	5	Merge to produce third sequence made up of $(N-1)^2$ original sequences	
18	25 1^5	25 1^4	25 1^3	 1^2	5 1^1	5	Distribute N-1 sequences	This cycle distributes the N-1 sequences in the same manner as they were distributed N-3 cycles prior to it (in cycle 15).
	25	25	25		5	5 5	Merge to Nth area	

Figure 8. Example of Crisscross Direct Access Distribution Technique (Part 2 of 3)

Distribution Cycle Number	Work Areas and Sequence Arrangement						Operation Performed	Comments (if any)
19	25	25 1^5	25 1^4	1^3	5 1^2	5 5 1^1	Distribute N-1 sequences	Each of the next N-3 cycles distributes the N-1 sequences in the same manner as described for cycles 2 through 5 above.
	25 5	25	25		5	5 5	Merge to Nth area	
20	25 5 1^1	25	25 1^5	1^4	5 1^3	5 5 1^2	Distribute N-1 sequences	
	25 5	25 5	25		5	5 5	Merge to Nth area	
21	25 5 1^2	25 5 1^1	25	1^5	5 1^4	5 5 1^3	Distribute N-1 sequences	
	25 5	25 5	25 5		5	5 5	Merge to Nth area	
	25	25	25	25		5	Merge to produce fourth sequence made up of $(N-1)^2$ original sequences	
22	25	25 1^5	25 1^4	25 1^3	1^2	5 1^1	Distribute N-1 sequences	This cycle distributes the N-1 sequences in the same manner as they were distributed N-3 cycles prior to it (in cycle 19).
	25 5	25	25	25		5	Merge to Nth area	
23	25 5 1^1	25	25 1^5	25 1^4	1^3	5 1^2	Distribute N-1 sequences	Each of the next N-3 cycles distributes the N-1 sequences in the same manner as described for cycles 2 through 5 above.
	25 5	25 5	25	25		5	Merge to Nth area	
24	25 5 1^2	25 5 1^1	25	25 1^5	1^4	5 1^3	Distribute N-1 sequences	
	25 5	25 5	25 5	25		5	Merge to Nth area	
25	25 5 1^3	25 5 1^2	25 5 1^1	25	1^5	5 1^4	Distribute N-1 sequences	
	25 5	25 5	25 5	25 5		5	Merge to Nth area	
	25	25	25	25	25		Merge to produce fifth sequence made up of $(N-1)^2$ original sequences	Second base level attained

[1]Indicates the first work area to receive a sequence during the distribution cycle.
[2]Indicates the second word area to receive a sequence during the distribution cycle.
[3]Indicates the third work area to receive a sequence during the distribution cycle.
[4]Indicates the fourth work area to receive a sequence during the distribution cycle.
[5]Indicates the fifth work area to receive a sequence during the distribution cycle.

Figure 8. Example of Crisscross Direct Access Distribution Technique (Part 3 of 3)

BALANCED TAPE TECHNIQUE

The sort phase distributes sequences onto half of the tapes used for intermediate storage. (If there is an odd number of tapes, the number that receives sequences is one more than the number that do not.) The sequences are placed onto successive tapes. That is, all tapes receive one sequence each before any receives a second sequence, and so on. All sequences produced by the sort phase are in the same order (i.e., ascending or descending), and the order is opposite to that desired for output.

Since the sort/merge program writes forward onto tape and reads backward from tape, successive passes of the intermediate merge phase reverse the order of all sequences.

The intermediate merge phase merges one sequence from each tape that received sequences in the sort phase. A single sequence from each of the input tapes is combined to form a single output sequence. The output sequences are written successively onto those intermediate storage tapes that were left empty in the sort phase. This is repeated until all input sequences have been processed. When all sequences from the input tapes are merged onto the output tapes, the tapes used for input and output are alternated. The merging process continues until the number of sequences is less than or equal to the merge order. At that time, the final merge phase merges the remaining sequences into a single sequence on the output device.

For fixed- or variable-length records, at the end of the sort phase and at the end of each pass of the intermediate merge phase, sort/merge checks to determine if two or less passes remain. If two intermediate passes remain, the program tests to see if the final sequences will be in the desired order. If the sequences will not be in the desired order, the output of the next pass is blocked in reverse order. At the end of that pass, the tapes are rewound and then read forward during the next pass with normal deblocking.

For fixed-length records, if at the end of the sort phase the number of sequences is less than or equal to the merge order and these sequences are found to be in reverse order, the work tapes are rewound. The final merge phase is then performed by deblocking the records from the back of the buffer to the front to produce desired sequences, thus eliminating the copy pass. However, for variable-length records it is not feasible to deblock the buffers in reverse. Therefore, a copy pass is required.

Figure 9 shows an example of the balanced tape technique. The output is to be in ascending order. The sort/merge program estimates an even number of intermediate merge passes. Therefore, the sort produces sequences in descending order (i.e., opposite to the output order).

The sort phase distributes successive sequences onto tapes A, B, and C. A receives the first, fourth, seventh, tenth, and thirteenth sequences; B the second, fifth, eighth, eleventh, and fourteenth; and C the third, sixth, ninth, and twelfth.

Tapes A, B, and C are input for the first intermediate merge phase pass; tapes D, E, and F are output. The intermediate merge phase merges one sequence from each input tape onto successive output tapes. Because the sort/merge program reads backwards from tape, the sequences are merged and distributed as follows: sequences 12, 13, and 14 are merged into one sequence and placed onto D; sequences 9, 10, and 11 are merged and placed onto E; sequences 6, 7, and 8 are merged and placed onto F; sequences 3, 4, and 5 are merged and placed onto D; and sequences 1 and 2 are merged and placed onto E. The sequences are now in ascending order.

Since the total number of sequences on D, E, and F (five) is greater than the merge order (three), another intermediate merge phase pass is required.

The input and output tapes are switched and the merging process continues. Sequences 1 through 8 are merged into one sequence and placed onto A. Sequences 9 through 14 are merged and placed onto B. The resulting sequences are in descending order.

When the total number of sequences is less than the merge order, the final merge is executed. The final merge phase combines the remaining two sequences into one sequence and places it onto the output device. The resulting sequence is in ascending order, the order desired for the output.

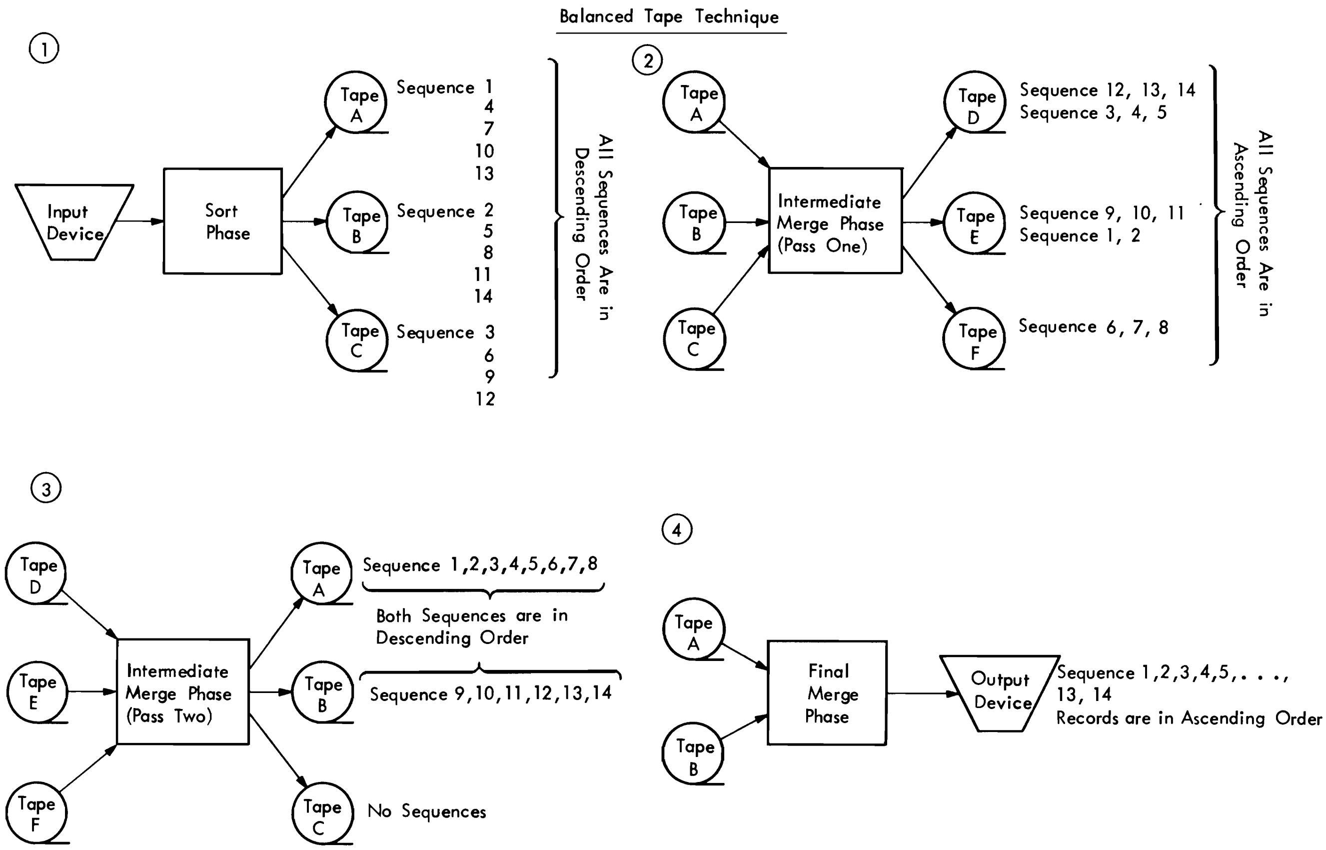


Figure 9. Example of Balanced Tape Sequence Distribution Technique

POLYPHASE TAPE TECHNIQUE

The sort phase distributes sequences onto all but one of the tapes used for intermediate storage. The order of the sequences on each tape alternates between ascending and descending. The sequences are distributed on the tapes in such an order that, when processed during the intermediate merge phase, the sequences on the tapes are depleted in a predetermined sequence.

The order of the first sequence on the first tape that receives a sequence is the same as the order desired for the output. The order of the first sequence on the other tapes is the opposite of the order desired for the output.

The intermediate merge phase combines sequences into longer sequences and places them onto the empty tape, until all the sequences on one tape are exhausted. That tape is now considered empty and can receive sequences from the other tapes, including the tape which was previously being used as output. The merging process continues until one sequence is left on each tape except one.

In the polyphase technique, the sequences are produced as directed by the sort algorithm. If a copy pass is required, it is used only for the last unordered sequences rather than the whole file.

The final merge phase combines the remaining sequences into a single sequence and places it onto the output device. Since the sort/merge program writes forward onto tape and reads backward from tape, each time sequences are merged their order is reversed. The input to the final merge phase is such that a sequence in the desired order is produced.

Example of Sequence Distribution in Polyphase Tape Technique

For clarification, the following text describes a simplified method for the sequence distribution algorithm used in the polyphase technique. In the example below, four tapes are available for intermediate storage. Each of the resulting sets of numbers represents a distribution of sequences by the sort phase.

Since there are four tapes, three of the tapes (W, X, and Y) can receive sequences from the sort phase.

Initially, an N-by-N identity matrix is formed for the N tapes receiving sequences from the sort phase. Shown below is the 3-by-3 identity matrix formed for the three tapes W, X, and Y.

1	0	0
0	1	0
0	0	1

The N numbers in the column for each tape are added together to find the first sequence-distribution set. The resulting set of numbers forms part of the column for each tape. Shown below is the addition resulting in the set of numbers 1, 1, 1.

1	0	0
0	1	0
0	0	1
1	1	1

The results of the addition are now considered part of the matrix. The last N numbers in the column for each tape are added together to find the second set of sequence distribution. Shown below is the addition resulting in the set of numbers 1, 2, 2.

1	0	0
0	1	0
0	0	1
1	1	1
1	2	2

The third set of sequence distribution is obtained in similar manner, as shown below.

1	0	0
0	1	0
0	0	1
1	1	1
1	2	2
2	3	4

Additional sets of sequence distribution are obtained in the same manner. Shown below are the first eight sets of sequence distribution obtained from the original 3-by-3 identity matrix.

1	1	1	*
1	2	2	
2	3	4	
4	6	7	
7	11	13	*
13	20	24	
24	37	44	
44	68	81	

The sets that contain only odd numbers (*) are not used. The remaining sets are shown below.

1	2	2
2	3	4
4	6	7
13	20	24
24	37	44
44	68	81

These sets each have one odd number. The odd number in each set is associated with the first tape (Y) to receive a sequence by rotating the numbers in some sets. The resulting sets are shown below.

W	X	Y
2	2	1
4	2	3
4	6	7
20	24	13
44	24	37
44	68	81

Each of these sets of sequence-level numbers represents a distribution of sequences that, when used as input to the intermediate merge phase, will result in an efficient merging process. For instance, if the sort phase produced nine output sequences, it would place four on W, two on X, and three on Y. If the number of output sequences is not equal to the sum of the numbers in any set, the tapes are assigned dummy sequences by using counters until the total of actual and dummy sequences distributed equals the sum of the numbers in a set. The intermediate merge phase then considers each dummy sequence as if it were an actual sequence in setting up a pass. When a dummy sequence is encountered during a merge pass, it is dropped if it is to be merged with an actual sequence. However, if all the sequences to be merged are dummies, a dummy sequence is carried to the next level.

Figure 10 illustrates the use of the polyphase technique. Tapes W, X, Y, and Z are used as intermediate storage devices. In this figure, the sort phase distributes 14 sequences onto W, X, and Y according to the set of sequence level numbers computed above. Since the total number of sequences (14) is not equal to the total of the numbers in any set, dummy sequences are assigned to X and Y until the total number of actual and dummy sequences equals the total of the next larger number (17) in the set 4, 6, 7. The A indicates an ascending sequence, D indicates a descending sequence, and d indicates a dummy sequence.

Polyphase Technique

W	X	Y	Z
A(3)	A(2)	D(1)	
D(5)	D(4)	A(6)	
A(7)	A(11)	D(8)	
D(9)	D(13)	A(10)	
	d	D(12)	
	d	A(14)	
		d	

Figure 10a. Sequences Produced by Sort Phase

W	X	Y	Z
	A(2)	D(1)	A(9)
	D(4)	A(6)	D(7,14)
		D(8)	A(5,13,12)
			D(3,11,10)

Note: Dummy sequences dropped.

Figure 10b. Sequences Merged Onto Z by Intermediate Merge Phase

W	X	Y	Z
A(4,8,3,11,10)		D(1)	A(9)
D(2,6,5,13,12)			D(7,14)

Figure 10c. Sequences Merged Onto W by Intermediate Merge Phase

W	X	Y	Z
A(4,8,3,11,10)	A(2,6,5,13,12 1,7,14)		A(9)

Figure 10d. Sequences Merged Onto X by Intermediate Merge Phase

SORTOUT
D(4,8,3,11,10,2,6,5,13,12,1,7,14,9)

Figure 10e. Sequences Merged Onto SORTOUT by Final Merge Phase

Note: Each term (e.g., A(3), D(3,11,10), A(4,8,13,11,10)) in Figure 10 represents a single sequence. Each number within parentheses in Figure 10a represents an original sequence (i.e., a sequence produced by the sort phase) and indicates the order in which that sequence was produced and distributed by the sort phase. (The numbers in Figure 10a do not represent records or control field values.) Two or more numbers within the parentheses of a term indicate that the term represents a sequence that was formed by a merge operation; in this case, the numbers in parentheses identify the original sequences (excluding dummy sequences) that were combined via one or more merge operations to form the sequence represented by the term.

Figure 10. Example of Polyphase Tape Sequence Distribution Technique

Descending order is desired for the final output. Therefore, the first work unit (Y) receives its initial sequence in descending order, and W and X receive their initial sequences in ascending order.

The last sequences from W, X, and Y are combined into one sequence of the opposite order which becomes the first sequence on Z (D9, d, and d from W, X, and Y, respectively, go as A(9) to Z). This is repeated until W becomes empty and available to receive sequences as shown in Figure 10b.

The intermediate merge phase combines the sequences as shown in Figures 10b, 10c, and 10d until W, X, and Z each have one sequence, and Y has no sequences.

The final merge phase merges the remaining three sequences from W, X, and Z and combines them onto SORTOUT (the output tape) in descending order. (See Figure 10e.)

Tables Used to Compute Sequence-Level Numbers

The actual implementation of the polyphase technique is accomplished by using three tables in the PPI area (IERRCA). (Refer to Section 5.) Module IERRCA is the phase-to-phase information (PPI) area that resides in storage throughout the sort program. This area serves as temporary storage and is used to communicate information among the various segments of the program. The three tables, with displacement in decimal form, are:

PPITPTBL+34
PPITPTBL+102
PPITPTBL+68

The functions of these tables vary from phase to phase. Following is a description of their functions for each phase.

Sort Phase (Module IERROJ)

Table	Purpose
PPITPTBL+34	Contains sequence counters for each physical unit. Each counter is incremented by one each time a sequence has been written on that physical unit.
PPITPTBL+102	Contains the total number of sequences required on each physical unit at the next higher level. These numbers are used to determine whether or not a unit should be used as the output device for the next sequence by comparing these numbers against the sequence counters in table PPITPTBL+34.
PPITPTBL+68	Contains the number of sequences for each logical unit. The next sequence level is calculated in this table after each level has been reached, moved to PPITPTBL+102, and then rotated to place the odd numbered sequence count on the desired unit.

Intermediate Merge Phase (Module IERROS)

Table	Purpose
PPITPTBL+34	Contains the sequence counters for each physical unit from the sort phase. Each counter is decremented by one each time a sequence is merged from the unit and incremented by one each time a sequence is merged onto the unit.
PPITPTBL+102	Contains the counters that maintain proper order for using the sequence-level numbers.
PPITPTBL+68	Contains numbers (that have been computed in table PPITPTBL+102 during the sort phase) that represent the difference between table PPITPTBL+34 and table PPITPTBL+102. These numbers are used to determine whether or not a given tape should be used as an input device for this intermediate merge operation.

Final Merge Phase

Table	Purpose
PPITPTBL+34	This table is used to determine the input units for the final merge phase.

Tables PPITPTBL+102 and PPITPTBL+68 are not used in this phase.

Example of Table use: The use of these tables for computing the sequence-level numbers is best illustrated in an example of a four-tape, 3-way merge.

Logical Tapes	A	B	C	D
Original Sequence-Level Numbers	1	1	1	

The number associated with the rightmost work unit (C) is added to each of the other (N-2) numbers to form the new sequence-level numbers. First the rightmost number is saved, and then the following takes place:

rightmost + B replaces C
rightmost + A replaces B, and finally
rightmost replaces A

Logical Tapes	A	B	C
Resultant Sequence-Level Numbers	1	2	2

The resultant sequence-level numbers are then rearranged so that the odd number of sequences is always associated with the same physical unit.

Logical Tapes (PPITPTBL+68)	A	B	C
	1	2	2
Physical Tapes	X	Y	Z
	2	2	1

If the above computations are done again on the old resultant sequence-level numbers (1, 2, 2), the resultant sequence-level numbers are:

Logical Tapes	A	B	C
Resultant Sequence-Level Numbers	2	3	4

The resultant sequence-level numbers are then rearranged so that the odd number of sequences is always associated with the same physical unit.

Logical Tapes (PPITPTBL+68)	A	B	C
	2	3	4
Physical Tapes	X	Y	Z
	4	2	3

The sequence-level numbers require updating when a working counter set in IERROJ equals zero. The value of this counter is obtained from a multiplication of the largest of the old sequence numbers in PPITPTBL+102 times the merge order minus one. The result of this multiplication is the number of sequences to be written before the sequence-level numbers are updated again. The counter is decremented by one each time a sequence is produced.

Example of 3-Way Merge

Old Sequence-Level Numbers Before Update	1	2	2
Sequence-Level Numbers After Update	2	3	4
Number of Sequences to Be Written Is	2(2)	= 4	

This summation is done twice when the sequence-level numbers are all odd. To determine whether the numbers are all odd, a counter is initially set to the merge order plus one. This counter is decremented by one each time a sequence level is reached. When this counter equals zero, all sequence-level numbers are odd. The action at this time is to:

- Reset the counter to the merge order +1.
- Update the sequence-level numbers again.
- Decrement the counter by one.
- Continue sorting.

OSCILLATING TAPE TECHNIQUE

The oscillating tape technique requires fewer passes over the data than do polyphase or balanced tape techniques discussed above. For optimum efficiency, this technique uses tape switching or read-while-write capabilities, when available.

The oscillating tape technique begins with the sort phase developing sequences of records on all but one of the intermediate work tapes. After the sort has created one sequence on each of the N-1 tapes, the sort goes to the merging operations immediately. Thus, the oscillating technique integrates the sort phase with the intermediate merge phase.

The N-1 sequences are read backwards and merged onto the available Nth tape. The N-1 tapes are then at load point, and control is passed to the internal sort. The next sequence is written onto tape N. The next N-2 sequences are written onto the next N-2 tapes. Control is again transferred to the merging portion of the program, and these N-1 sequences are merged onto the remaining available merge tape.

This process is continued until each of N-1 tapes has had N-1 sequences merged onto it; i.e., the oscillating sort has created N-1 sequences from $(N-1)^2$ sequences. At this point, the N-1 sequences are merged onto the available tape. The process repeats until another tape contains a sequence formed from $(N-1)^2$ sequences.

When each of N-1 tapes contains a sequence formed from $(N-1)^2$ original sequences, these sequences are again merged onto the available tape which now contains a sequence formed from $(N-1)^3$ original sequences. This iterative process continues until all the input records have gone through the sort. A partial merging pass may be required, followed by a final merge operation onto the output tape. Partial merges are performed as necessary to reduce the number of remaining sequences to N-1 or fewer. Figure 11 illustrates the technique using 27 sequences with four work tapes.

The selection of the work tapes onto which the N-1 sequences are distributed during each distribution cycle is determined by the following method:

1. The first cycle distributes a sequence onto each of N-1 work tapes. An example of this is distribution cycle number 1 in Figure 11.

2. Each subsequent cycle distributes: (a) the first of the N-1 sequences onto the work tape that did not receive a sequence during the previous cycle; (b) the second of the N-1 sequences onto the work tape that received the first sequence during the previous cycle; (c) the third of the N-1 sequences onto the work tape that received the second sequence during the previous cycle. For example, see distribution cycles 2 through 9 in Figure 11.

Thus, in this method, sequences are distributed onto the work areas in groups of N-1 or three per distribution cycle in the order: ABC, DAB, CDA, BCD, ABC, DAB, CDA, BCD, ABC.

Distribution Cycle Number	Work Tapes and Sequence Arrangement				Operation Performed
	A	B	C	D	
1	1^1	1^2	1^3		Distribute N-1 sequences
				3	Merge to Nth tape
2	1^2	1^3		3 1^1	Distribute N-1 sequences
			3	3	Merge to Nth tape
3	1^3		3 1^1	3 1^2	Distribute N-1 sequences
		3	3	3	Merge to Nth tape
	9				Merge to produce a sequence made up of $(N-1)^2$ original sequences
4	9	1^1	1^2	1^3	Distribute N-1 sequences
	9 3				Merge to Nth tape
5	9 3 1^1	1^2	1^3		Distribute N-1 sequences
	9 3			3	Merge to Nth tape
6	9 3 1^2	1^3		3 1^1	Distribute N-1 sequences
	9 3		3	3	Merge to Nth tape
	9	9			Merge to produce second sequence made up of $(N-1)^2$ original sequences
7	9 1^3	9	1^1	1^2	Distribute N-1 sequences
	9	9 3			Merge to Nth tape

Figure 11. Example of Oscillating Tape Sequence Distribution Technique (Part 1 of 2)

Distribution Cycle Number	Work Tapes and Sequence Arrangement				Operation Performed
	Tape 1	Tape 2	Tape 3	Tape 4	
8	9	9 3 1[1]	 1[2]	 1[3]	Distribute N-1 sequences
	9 3	9 3			Merge to Nth tape
9	9 3 1[1]	9 3 1[2]	 1[3]		Distribute N-1 sequences
	9 3	9 3		3	Merge to Nth tape
	9	9	9		Merge to produce third sequence made up of $(N-1)^2$ original sequences
				27	Merge to produce a sequence made up of $(N-1)^3$ original sequences

[1]Indicates the first work tape to receive a sequence during the distribution cycle.
[2]Indicates the second work tape to receive a sequence during the distribution cycle.
[3]Indicates the third work tape to receive a sequence during the distribution cycle.

Figure 11. Example of Oscillating Tape Sequence Distribution Technique (Part 2 of 2)

Merging Technique

The sort/merge program uses a binary- compare network as the merge technique to merge the input to (1) successive passes of the intermediate merge phase for sorting applications and (2) the final merge phase for both sorting and merging applications.

In performing the merge operations, an input record is inserted in proper sequence within an already ordered sequence. The insertion of the input record forces out the next output record. (See Figure 12.) In the intermediate and final merge phases, fixed-length and variable-length unspanned records are moved directly from the input buffers to the output buffers. Records which are VRE records for output are handled somewhat differently in the final merge phase. These records pass through a work area before being moved to the output buffers by the data management PUT routine in move mode. In either case, only the record addresses are manipulated in forming sequences.

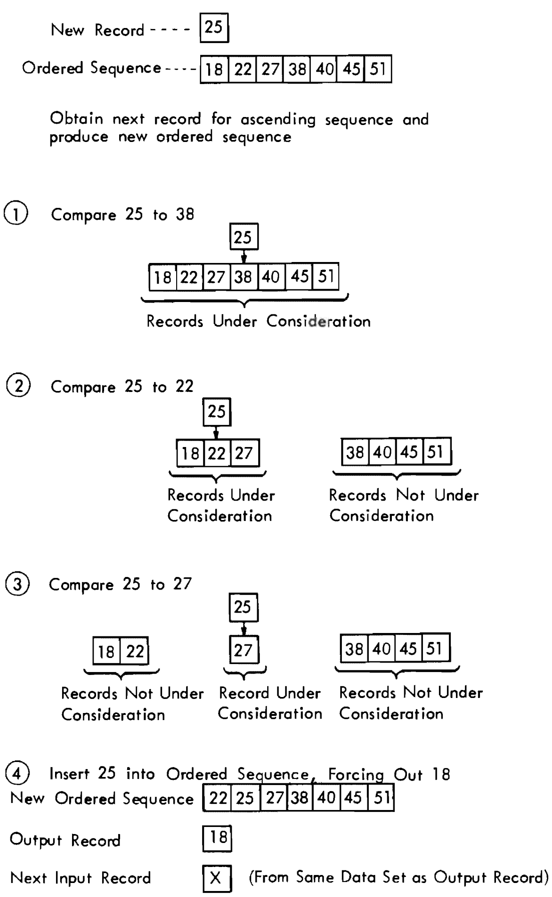


Figure 12. Merging Technique Using Binary-Compare Network

The first step is to compare the new record to the middle record in the ordered sequence to determine if the new record collates higher or lower. If it collates higher, all lower records are eliminated from consideration. If it collates lower, all the higher records are eliminated.

The new record is then compared to the middle record of the remaining records under consideration to determine in which half of these records the new record belongs. The process continues until there is one record remaining. The new record belongs on either one side or the other of this record, depending upon how the new record collates. The records on one side of the newly inserted record are shifted one position and the new lowest record is then moved to the output buffer.

INTERMEDIATE MERGE PHASE

The following is a discussion of intermediate merge phase processing for the balanced tape technique.

Initially, one block of records from each data set to be merged is brought into main storage. The relative collating order of the first record from each sequence is determined, and an ordered table of their addresses is set up. The lowest of these determines the first record to be written. After a record is selected for output, another record is taken from the same block that contained the output record. The new record's sequencing relative to the other records is determined using the binary-compare network.

The process continues until an end-of-sequence is reached. The merge order is then reduced by one and the merging process continues. When the end of all the sequences is reached, the merge of the input sequences is complete. The output data set now contains a complete sequence and an end-of-sequence indicator is written.

The merge network is reset to its original value for the merge order, the process is initialized again by placing the first block of records from each new input sequence into main storage, and the merging process continues.

Output sequences are produced in this manner until an input end-of-file is reached from one of the input data sets. At this point the merge order is again reset to its original value minus the number of input data sets that have reached end-of-file.

When all input data sets reach end-of-file, an entire pass of the intermediate merge phase is completed, and the merge order is reset to its original value for the next pass.

FINAL MERGE PHASE

The operation of the binary-compare network in the final merge phase is identical to the intermediate merge phase, except that there is only one sequence on each input data set.

Record Movement in the Sort/Merge Program

The sort/merge program contains several logical points at which records are moved from one location to another. Figure 13 illustrates the movement of records from the time they are read from the input device until they are written out on the output device.

MODULES USED FOR RECORD MOVEMENT

The sort/merge program uses the input/output modules (read, write, block, and deblock) to control the flow of records. The input/output modules also control the operational overlap of channel processing with CPU operations, thus providing efficient utilization of system resources. In performing these functions, the input/output modules communicate with the data management area of the control program, with the input/output devices used for intermediate storage, and with other sort/merge modules.

For the initial input to the sort/merge program and for the final output, the input/output modules use the data management Queued Sequential Access Method (QSAM) routines, using the GET and PUT macro instructions. Locate mode is used for fixed or variable unspanned records. For spanned variable records, an extra work area is required and move mode is used. This permits device independence between the sort/merge program and the user's data sets. All other input/output activity within the sort/merge program uses the Execute Channel Program (EXCP) macro instruction. The read and write modules used in this connection are device dependent, since each module is oriented to a particular type of intermediate storage device.

The particular read, write, block, and deblock modules used in any given application depend on the intermediate storage device, the form of the records (fixed or variable) being processed, the type of application (sorting or merging), and the presence of user modifications.

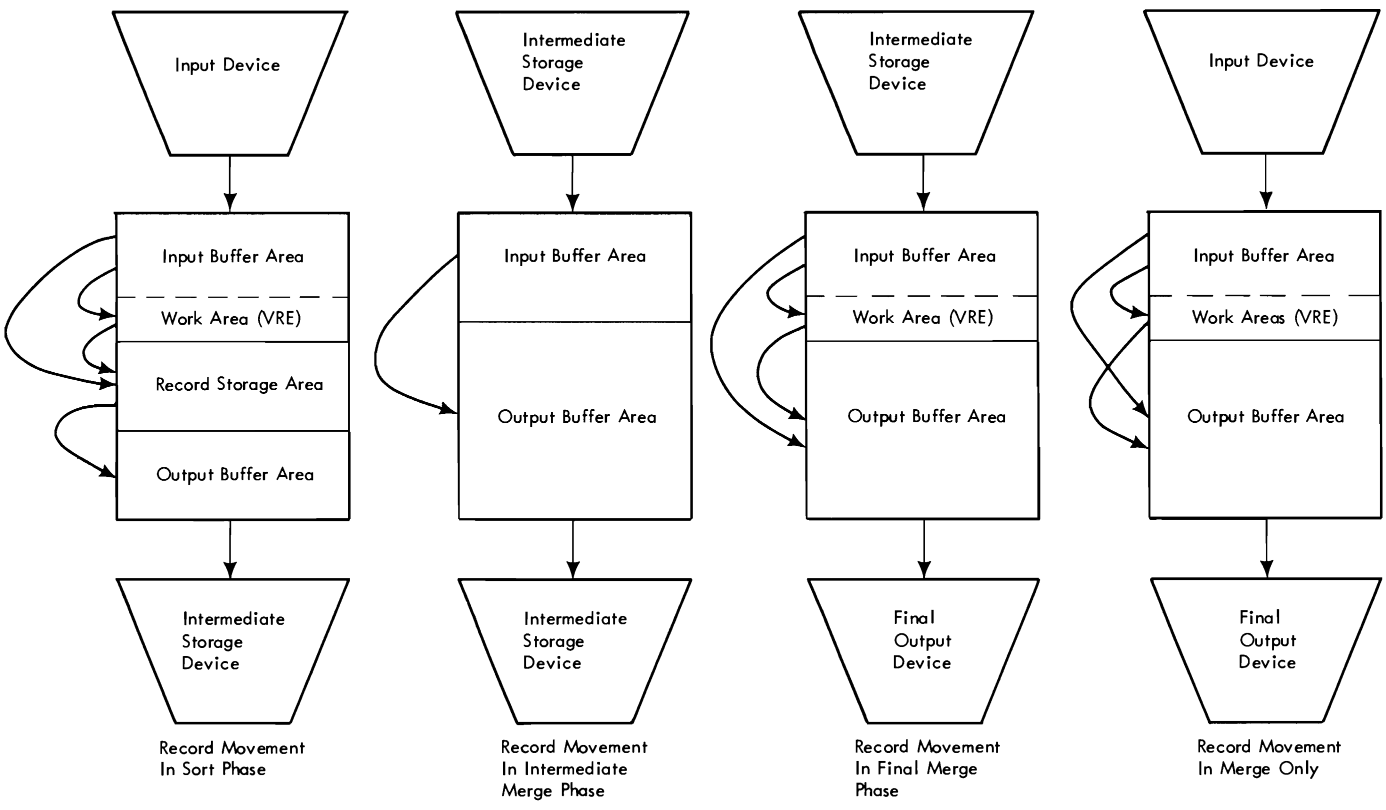


Figure 13. Movement of Records in the Sort/Merge Program

In general, the assignment program modules perform initialization functions, such as setting up areas, setting counters, and modifying instructions to adapt the running programs to a particular application. The input/output functions begin by establishing buffer areas and opening the data sets. This is done before the actual running program for each phase.

In the discussion that follows, consideration is given only to the running program modules, those modules that perform the actual sorting and merging. The input to the sort phase of a sorting application and the input to a merging application rely on the data management QSAM routines to fill the input buffers and to deblock records. The intermediate merge phase and final merge phase for a sorting application use the EXCP macro instruction to place records from the input data sets into buffers and to prime (initially fill) the merging network areas. Deblocking of records in the intermediate merge and the final merge for a sorting application is done by the sort/merge program.

In the sort phase, the deblock modules move records from the input buffers to the RSA. Variable spanned records are treated differently. The deblock modules move them from the input buffers to a work area and then into the RSA. The records remain in the RSA during the processing performed by the sorting modules. To identify the records being sorted, the deblock modules provide the RSA addresses of the records to the sorting modules.

In the intermediate merge phase and the final merge phase, the records are sequenced while they remain in the input buffer area. Part of the buffer-filling function of these two phases is to assign an increment (value) to each input data set and to the first record taken from each data set. This increment is used to identify the data set from which to select new records. This is the same data set that provided the current output record.

The deblock modules used in these phases provide the merging modules with the buffer addresses of the records. The increments are placed in the high-order byte of the record addresses and serve as keys for identification of the records.

The block/deblock modules keep count of the records in the input data sets as they are brought into the sort/merge program. At the end of a phase, a comparison is made between the record count in the phase-to-phase information area and the current value as recorded by the block/deblock module being used. The effects of user modifications that insert or delete records are not included in the comparison. If the comparison is unequal, the message IER047A is printed and the sorting operations terminate.

In the intermediate merge and in the final merge, deblock modules determine when the input buffers are empty. When a buffer is empty, a deblock module gives control to a read module to refill the buffer.

After the input areas have been filled, the modules that control the ordering of records begin to produce record sequences. As each output record is determined, the ordering module gives the record address to a blocking module which moves the output record to an output buffer. In the final merge phase, VRE records pass through a work area before being moved to the output buffer. The block modules also give the address of each current winner record to the deblock module so that another record from the same input area can be passed to the ordering routine for sequencing.

When an output block of records is completed and ready to be written on the output data set, the block modules transfer control to the write modules in both the sort phase and the intermediate merge phase.

For applications using tape for intermediate storage, the ordered records are placed in the output buffers in such a manner that the first record is located at the numerically highest main storage locations in the buffer. The second record is then placed on the low-address side of the first record. Continuing in this manner, the output buffers are constructed in a high-to-low direction. Thus, when the write modules place the records on the intermediate storage device, these records appear in a backward arrangement. This technique of putting the records in ascending and descending orders is implemented to process records in a proper manner for reading backwards.

When the records are read backward, they appear in reverse order. For each subsequent pass through the intermediate merge phase of the sort/merge program, the ordering of the output sequences is always opposite to the ordering of the previous pass. This read-backwards procedure eliminates the need for rewinding the tape.

For direct access applications the records are placed in order from low to high main storage locations.

In the final merge and for a merging application, the actual blocking of records is done by data management according to the format specified on the SORTOUT dd statement. The block routine moves the record to an output buffer specified by the QSAM PUT macro instruction (locate mode) if fixed or variable unspanned records are used. If variable spanned records are used, they are moved to a work area by the block routine and from the work area into an output buffer by the QSAM PUT macro instruction (move mode).

RECORD MOVEMENT TECHNIQUES

The various record-movement techniques, the conditions determining their selection, and the RSA structure for fixed- and variable-length records are discussed in the following sections.

Fixed-Length Records

There are two types of fixed-length record moves: the in-line move and the multiple move.

The in-line move consists of a single move instruction appropriate to the record length for the particular application and is used whenever:

- Record length is 256 bytes or less; and
- No user modifications are present.

The multiple move consists of a set of instructions (called the move list) and is used to move each complete record. The move list moves records in increments of 256 bytes. If the record length is not a multiple of 256 bytes, the final move instruction moves that portion of the record remaining after the last 256-byte move. The multiple move is used whenever:

- Record length is greater than 256 bytes; or
- User modifications to the records are present.

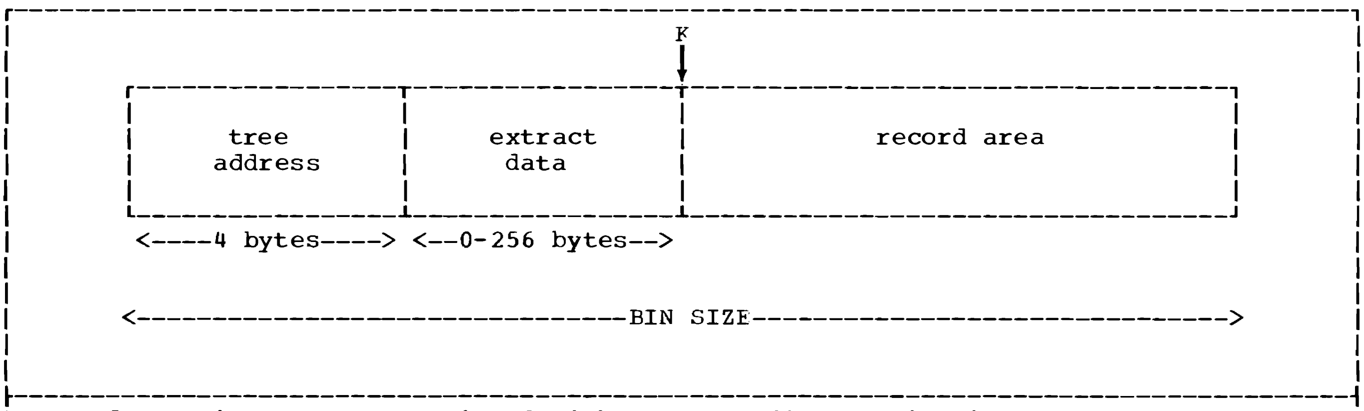
FIXED-LENGTH RECORDS IN THE SORT PHASE: When fixed-length records are placed in the RSA for the sort phase, they are placed in sections called bins. Each bin has the same size: this size is calculated by the definition phase. The calculated bin size is placed in the phase-to-phase information area, which also contains the starting address of the RSA.

For fixed-length records, a bin in the RSA has the format shown in Figure 14.

To determine the locations of records in the RSA, a deblock assignment module assigns a value K to the starting address of the first record in the RSA. (See Figure 14.)

K = RSA start address + Bin size - Record Length

In this equation, the record length is rounded off to the next full word if it is not already in multiples of a full word. K represents the initial address that is given to the sorting module by the running deblock module.



Tree address is a value associated with the record's location in the sort network tree structure. (Refer to Sorting Techniques.) The tree address occupies four bytes and begins on a full-word boundary.
Extract data is from zero bytes to 256 bytes in length. When the extract module is not used, it is zero bytes. When the extract module is used, it is from 4 to 256 bytes and contains the extracted control fields of the record.
K represents the starting address of the first record in the RSA.
The complete fixed-length record occupies the remainder of the bin. The record itself must begin on a full-word boundary.

Figure 14. Fixed-Length Record Format in the Record Storage Area

The RSA may not be in contiguous locations. Hence, when incrementing by bin size to get the address of the next record, a check for the end of the current RSA segment must be made. If the end is reached, the starting address of the next RSA segment becomes the address of the next record.

FIXED-LENGTH RECORDS IN THE MERGE PHASES: In the intermediate merge, the final merge, and the merge only operation, the move list is used to transfer records directly from input buffer to the output buffer.

To allow for any changes in record length due to user modifications in a preceding phase, the move list is generated in each phase by an assignment program.

Variable-Length Records

The sort phase has a separate move module to move variable-length records. This move module moves variable unspanned records from an input buffer to an RSA bin or from a bin to an output buffer. The module moves variable spanned records from an input buffer to a work area and then to an RSA bin, or from a bin to an output buffer.

In the two merge phases and the merge only phase, variable-length unspanned records are moved from the input buffer to the output buffer by the appropriate block/deblock module via a multiple move routine.

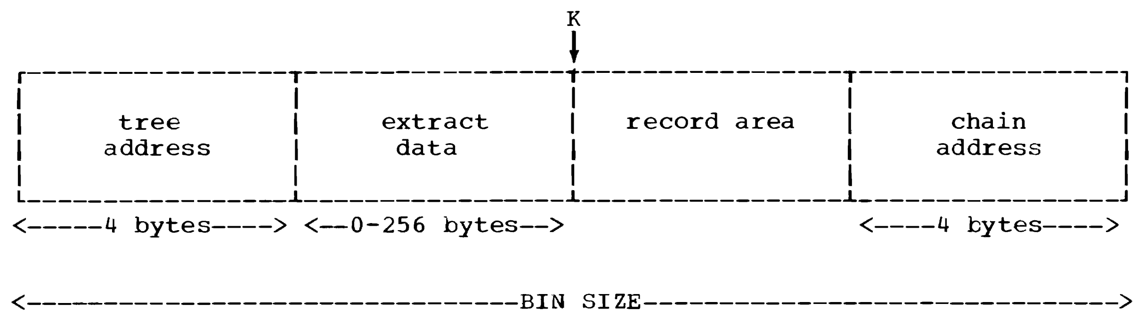
For spanned records, the same move routine is used in the intermediate merge phase, but in the final merge phase, the records are moved into a work area and the data management routine PUT in move mode is used to move them to the output buffer. In the merge only operation, the records are placed in the work area by the QSAM GET macro instruction in move mode and moved from there into a second work area. The PUT macro instruction in move mode moves them from the second work area to the output buffer.

VARIABLE-LENGTH RECORDS IN THE SORT PHASE: Variable-length records occupy bins in the RSA in a manner similar to that used for fixed-length records. The definition phase calculates a bin size for variable-length records so that the available main storage is most efficiently used. The bin size and the starting address of the RSA are placed in the phase-to-phase information area. A bin may contain all or part of one record.

For variable-length records, the initial bin for each record in the RSA has the format shown in Figure 15.

When a variable-length record requires more than one bin, the extra bins, called continuation bins, have the format shown in Figure 16.

The location of each continuation bin for a record is given in the chain address portion of the previous bin that contains



Tree address is a value associated with the record's location in the sort network tree structure. (Refer to Sorting Techniques.) The tree address occupies four bytes and begins on a full-word boundary.
Extract data is from zero bytes to 256 bytes in length. When the extract module is not used, it is zero bytes. When the extract module is used, it is from 4 to 256 bytes and contains the extracted control fields of the record.
K represents the initial address of the first record in the RSA.
Record area contains as much of the record as will fit in the bin after the other three areas have been accounted for; it must begin on a full-word boundary.
Chain address, which must begin on a full-word boundary, indicates where more of the record is located if the current bin does not contain the end of the record; otherwise this address is meaningless.

Figure 15. First Bin Format for Variable-Length Record in Record Storage Area

part of the same record. Each bin is always filled before a continuation bin is used. In the final bin, the record area may or may not be completely used. (See Figure 16.) The chain address in a bin for a variable-length record must begin on a full-word boundary; also, all continuation bins must begin on a full-word boundary.

If a variable-length record extends into two or more bins, the bin size must be large enough so that all control data fields (extracted or not) appear in the first bin. This is because the compare networks and routines are able to address only the first bin.

The RSA address of the beginning of the first record, K, (see Figure 15) is found from the equation:

K = RSA start address + (Bin size - 4) - (Length of the first bin record area)

The address constant indicating the beginning of each continuation bin for a record is found by adding BIN SIZE to the record storage area address of the preceding bin. This location is placed in the chain address area of the preceding bin. For example, if the RSA starts at location 12500, the extract occupies 12 bytes, the BIN SIZE is 64 bytes, and the first variable-length record is 150 bytes long, then the record will occupy parts of three bins as shown in Figure 17.

At the start of processing, a phase assignment program chains together all bins in the RSA, taking into consideration that RSA may not be in one contiguous piece.

If the variable-length record illustrated in Figure 17 was not the first record in the input data set but was placed into the record storage area some time after the initial filling of the record storage area, it might occupy three noncontiguous 64-byte bins as shown in Figure 18. (All bins are the same size, i.e., 64 bytes.)

When a record is written out from the bins in the RSA, the address of the next

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│                                                                                  │
│        ┌───────────────────────────────┬──────────────┬────────────────┐         │
│        │                               │              │                │         │
│        │          record area          │   a portion  │     chain      │         │
│        │                               │may be unused │    address     │         │
│        │                               │              │                │         │
│        └───────────────────────────────┴──────────────┴────────────────┘         │
│                                                       <----4 bytes---->          │
│                                                                                  │
│        <--------------------------------BIN SIZE------------------------------>  │
│                                                                                  │
│Record area, which must begin on a full-word boundary, contains as much of the record │
│as will fit in the bin after the chain address area has been accounted for.  Note that │
│a portion of the record area storage allotment may be unused if the end of the record │
│occurs before the beginning of the chain address.                                 │
│Chain address, which must begin on a full-word boundary, indicates where more of the │
│record is located if this bin does not contain the end of the record; otherwise this │
│address is meaningless.                                                           │
└──────────────────────────────────────────────────────────────────────────────────┘
```

Figure 16. Continuation Bin Format for Variable-Length Record in Record Storage Area

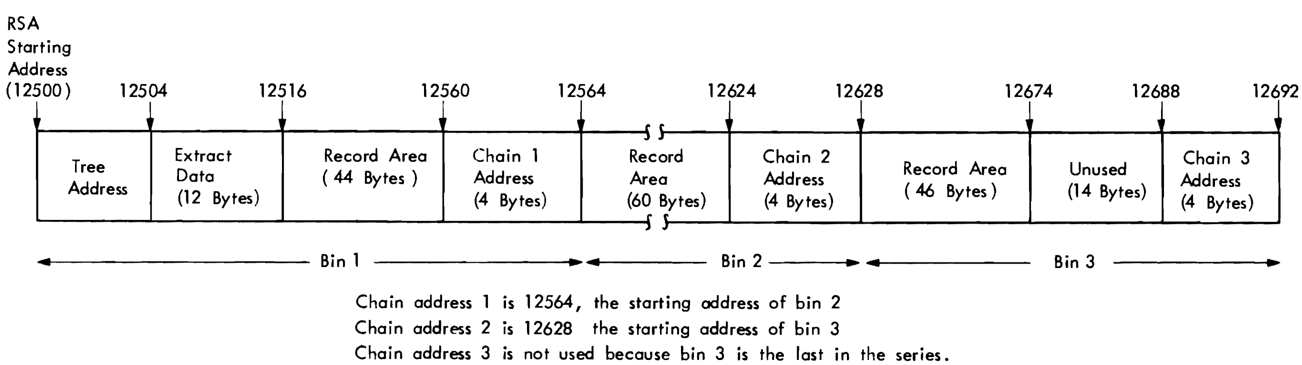


Figure 17. Example of a Variable-Length Record in Contiguous Bins in Record Storage Area

available bin, which has been saved in the PPI area, is placed in the last location of the bins just made free. The beginning address of the first bin is placed in location PPIBDSVA, which is also a part of the PPI. This address is maintained by the deblock and block modules used in the particular application. The bin addresses are given to the deblock module by the block module being used via PPI. New records are assigned bin locations from the available list. If a new record requires more bins than are available, the move is not completed until enough bins are made available by the block routine.

As a deblock module assigns bins to new records, it uses the chaining address to tie together bins that may be scattered throughout the RSA. The address of the first bin of each record is given to the sorting module.

The sort phase move module for variable-length records requires the:

- Address of the first bin of the record.
- Bin size used in the sort/merge application.
- Buffer address (for unspanned records).
- Work area address (for spanned records).
- Size of the record.
- Movement of the record, i.e., whether the records are to be moved from the input area to the RSA (deblocking) or from the RSA to the output buffer area (blocking). Note that for VRE records, record movement is as follows: from the work area to the RSA (deblocking) and from the RSA to the output buffer area (blocking).
- Indication by the deblock module whether or not this is a continuation of the previously incomplete move.

<u>VARIABLE-LENGTH RECORDS IN THE MERGE PHASES</u>: The buffer-to-buffer move in the final merge phase or merge only phase requires the location of the input buffer from which the record is to be moved and the location of the output buffer into which the record is to be moved. These moves are not in a separate module; they are coded in-line in the particular block/deblock module used. In the final merge phase, variable spanned records are moved from the input buffer to a work area and then to an output buffer. The locations of the buffers and the work area are required.

In a merge only phase, variable spanned records are moved from an input buffer to an input work area (one for each input data set) to an output work area (one only), and then to the output buffer. The locations of the buffers and the work areas are required.

```
13332     13336           13348                                        13392     13396
  ┌─────────┬───────────────┬─────────────────────────────────────────────┬─────────┐
  │         │               │                                             │  12692  │
  │  tree   │    extract    │                 record area                 │  chain  │
  │ address │     data      │                 (44 bytes)                  │ address │
  │(4 bytes)│  (12 bytes)   │                                             │(4 bytes)│
  └─────────┴───────────────┴─────────────────────────────────────────────┴─────────┘
  <---------------------------------------Bin 1------------------------------------->

12692                                                                  12752     12756
  ┌───────────────────────────────────────────────────────────────────────┬─────────┐
  │                                                                       │  12948  │
  │                              record area                              │  chain  │
  │                              (60 bytes)                               │ address │
  │                                                                       │(4 bytes)│
  └───────────────────────────────────────────────────────────────────────┴─────────┘
  <---------------------------------------Bin 2------------------------------------->

12948                                                  12994           13008     13012
  ┌──────────────────────────────────────────────────────┬────────────────┬─────────┐
  │                                                      │                │ end of  │
  │                     record area                      │     unused     │  chain  │
  │                     (46 bytes)                       │   (14 bytes)   │ address │
  │                                                      │                │(4 bytes)│
  └──────────────────────────────────────────────────────┴────────────────┴─────────┘
  <---------------------------------------Bin 3------------------------------------->
```

Figure 18. Variable-Length Record Not in Contiguous Bins of Record Storage Area

Section 3: Program Organization

This section describes the functions and structure of the sort system interface and each of the five phases that make up the sort/merge program. For a sorting application, all five phases are used, unless the intermediate merge phase can be bypassed, as explained under "Overall Flow of the Sort/Merge Program" in Section 1, and as shown in Chart 10. (Charts 10-73 are at the end of Section 3.) For merging applications, only the definition, optimization, and final merge phases are used.

At system generation time, four load modules are produced: two (IERRC000, alias SORT, and IERRCB) for the sort system interface and one each (IERRCM and IERRCZ) for the definition and the optimization phases.

For every sorting application, the object modules required for the sort, intermediate merge, and final merge phases are selected during the execution of that phase.

Provisions are contained in the sort, intermediate merge, and final merge phases for inclusion of user routines at program-modification exits. For the phase having the exit, the linkage editor includes the user routines in the load module. When proper specifications are provided on the MODS control statements, link editing does not occur. (See Appendix A for the various modification exits.) The data areas used by the assignment program are functionally related to all assignment program object modules and, therefore, reside in main storage during execution of the assignment program.

Sort System Interface

The two load modules (IERRC000 and IERRCB) in the sort system interface perform the following functions:

The IERRC000 module requests execution of the definition phase, linkage editor when required, and IERRCB.

The IERRCB module:

1. Requests execution of the optimization phase.
2. Loads IERRCV and branches to it.
3. Returns control to the control program upon termination of the sort/merge program.

Definition Phase

The definition phase consists of a control module (IERRCM) and a set of sequentially executed modules. (See Chart 20.)

The modules in this phase:

1. Read and interpret control statements.
2. Obtain information from the operating system's control blocks and tables (TIOT, UCB, JFCB, and DSCB).
3. Determine bin size for the record storage area.
4. Calculate the blocking factor (B) for intermediate storage devices and the number of records (G) that can be sorted at one time for sorting applications.
5. Determine if there is enough storage available for a merging application.
6. Produce lists of user routines to be link edited as specified on the MODS statement.

Optimization Phase

The optimization phase performs the calculations needed to optimize the execution of the sort/merge program for a given application.

This phase consists of a control module (IERRCZ) and a set of sequentially executed modules. (See Chart 30.) Where one of several modules is used, this phase determines which module to use, based on data in the phase-to-phase information area.

The modules in this phase:

1. Expand the phase-to-phase information area.
2. Obtain information from the operating system's control blocks and tables for a sorting application.
3. Store information about intermediate storage devices in the phase-to-phase information area for a sorting application.

4. Generate the extract module, if required. (The extract module is used to optimize record comparisons in the sort, intermediate merge, and final merge phases.) The extract module is required when modification exit E61 is used or if the control fields in the data records contain anything other than binary data on byte boundaries and/or character data.

5. Generate the equals module (similar in function to the extract module), if records contain more than one control data field and the extract module is not generated.

6. Optimize the usage of channel configuration in order to provide the maximum overlap of input/output operations.

Sort Phase

The sort phase (Charts 40 through 44) performs the initial ordering of the input records. This is a one-pass phase (that is, each record is processed only once) that arranges the records of the input data set into ordered sequences. The sequences are written on the intermediate storage devices according to a predetermined distribution procedure.

The sort phase contains a load module, assignment modules, and running modules. Initial entry to the sort is to the phase control module IERRCV. This module requests the execution of the appropriate sort phase routines as directed by IERRC6, and the load routine IERRC9 loads these modules into main storage with the phase-to-phase information area. Each of the assignment routines is brought in and executed by the load routine one at a time. After all the assignment routines have been executed, the load routine deletes the last assignment routine and branches to the first running program which is already loaded. The first running program deletes IERRC9 and actual processing of records is begun.

SELECTING MODULES FOR THE SORT PHASE

Module IERRC6 checks PPISW1 in the phase-to-phase information area to determine:

1. The type of device to be used for intermediate storage.

2. The sequence distribution technique chosen by the definition phase.

3. The presence of user modification routines for exits E15, E16, and E25.

4. Whether or not the input or output will contain spanned records.

IERRC6 sets bits in WSWITCH to indicate its findings. It then ORs WSWITCH into each entry in TBLPH1RN, a table of phase 1 running module masks. A result of all ones indicates that the module is needed for the sort/merge execution. IERRC6 stores the last three characters of each necessary running module in the phase-to-phase information area and sets up a load list of user modification routines for phase 1.

Assignment modules are selected in the same manner using TBLPH1AS, a table of assignment module masks, and a load list of the necessary assignment modules is created.

Referring to the load list and the phase-to-phase information area module list, IERRC9 loads the modules.

When messages are to be printed in the sort phase, the conditions are detected by the assignment and running modules and the actual printing is done by the control module IERRCV. If the sort is terminated for any reason, control is given to the phase control module which, in turn, returns control to the sort system interface module IERRCB.

The assignment modules initialize the sort phase. Their functions include:

1. Setting up the sorting tree structure, data control blocks, and input/output blocks.

2. Opening the data sets.

3. Modifying the running-program modules to adapt them for a specific application.

The actual processing of the records is performed by the running-program modules. These modules bring records into main storage, sort them into ordered sequences, and then write these sequences onto intermediate storage devices.

The main functions of the running programs in the sort phase are as follows:

1. Initiate input operations.

2. Deblock records and pass them to the ordering network.

3. Sequence the records using the replacement-selection technique.

4. Block the records when received from the ordering network.

5. Initiate output operations.

6. Close the files on detecting EOF.

7. Check record totals.

The sort phase after performing the initial processing of the records passes control to IERRCV which initiates the loading of the intermediate merge phase.

The input of records is controlled by the control program's data management routines, and the output is controlled by the sort/merge program. (Refer to "Record Movement in the Sort/Merge Program.") The sorting method used to order the records into sequences is a version of the replacement-selection technique. (Refer to "Sorting Technique.")

A set of decision tables (Charts 42 through 44) shows the sort phase modules used for a given application.

Intermediate Merge Phase

The intermediate merge phase (Charts 50 through 53) combines the short sequences produced by the sort into a lesser number of longer sequences. Each pass is capable of merging up to 16 previously sorted record sequences. This phase makes as many passes as necessary until the number of record sequences resulting from a given pass is equal to or less than the merge order.

The intermediate merge phase contains a load module, assignment modules, and running modules. Initial entry to the intermediate merge phase is to the phase control module IERRCV. This routine requests the execution of the appropriate routines as directed by IERRC7, and the load routine IERRC9 loads these modules into main storage with the phase-to-phase information area. Each of the assignment routines is brought in and executed by the load routine one at a time. After all the assignment routines have been executed, the load routine deletes the last assignment routine and branches to the first running program. At this time IERRC9 is deleted and actual processing of records is begun.

SELECTING MODULES FOR THE INTERMEDIATE MERGE PHASE

Module IERRC7 checks PPISW1 in the phase-to-phase information area to determine the characteristics of the sorting application and sets bits in WSWITCH to reflect the characteristics. It then ORs WSWITCH into each entry in TBLPH2RN, a table of phase 2 running module masks. A particular module is needed for sort/merge execution if the OR operation results in all ones. IERRC7 stores the last three characters of the module name of each required module in the phase-to-phase information area and prepares a load list of user modification routines to be used in phase 2.

Using TBLPH2AS, a table of phase 2 assignment module masks, IERRC7 selects the required assignment modules and places their names in a load list. IERRC7 branches to module IERRC9, which refers to the load list and the phase-to-phase information area module list, to load the modules.

When it is necessary for messages to be printed in the intermediate merge phase, such requests are passed from the running and the assignment modules to IERRCV which controls the actual printing. When this phase is terminated for any reason, control is given to the phase control module which, in turn, returns control to the sort system interface module IERRCB.

The assignment modules initialize the intermediate merge phase. Their functions include:

1. Setting up the buffer areas for the merge modules.

2. Generating data control blocks and input/output blocks.

3. Opening all the intermediate storage data sets.

4. Modifying the running-program modules to adapt them for a given application.

The actual processing of records is performed by the running-program modules. These modules bring records into main storage, merge the existing record sequences into longer sequences, and write the resulting sequences on the indicated intermediate storage devices.

The main functions of running programs in the intermediate merge phase are as follows:

1. Initiate input operations.

2. Deblock records and pass them to the merge network.

3. Sequence the records using the binary-insertion technique.

4. Block the records when received from the merge network.

5. Initiate output operations.

6. Close files on detecting EOF.

7. Check record totals.

8. Determine if another intermediate merge pass is required.

After reducing the number of sequences to less than or equal to the specified merge order, the intermediate merge phase passes control to IERRCV, which loads the final merge phase.

The input and output of records is controlled by the sort/merge program. (Refer to "Record Movement in the Sort/Merge Program.") The sequence distribution methods used to form the output record sequences are the balanced direct access technique, the crisscross direct access technique, the balanced tape technique, the polyphase tape technique, and the oscillating tape technique. (Refer to "Sequence Distribution Techniques.")

A set of decision tables (Charts 52 and 53) indicates the intermediate merge phase modules used for a given application.

When the oscillating tape or crisscross direct access sequence distribution techniques are used, control alternates between the sort phase which produces and distributes sequences and the intermediate merge phase which combines the sequences into longer sequences. See Charts 60 and 61.

Final Merge Phase

The final merge is a one-pass phase (Charts 70 through 73) and produces a single ordered record sequence, thus completing the sorting/merging process. In a sorting application, this phase follows the intermediate merge phase. In a merging application, this phase is executed immediately after the optimization phase.

The final merge phase contains a load module, assignment modules, and running modules. Initial entry to the final merge phase is to the phase control module IERRCV. This module requests the execution of appropriate routines for the final merge phase as directed by IERRC8, and the load routine IERRC9 loads these modules into main storage with the phase-to-phase information area. Each of the assignment routines is brought in and executed by the load routine one at a time. After all the assignment routines have been executed, the load routine deletes the last assignment routine and branches to the first running program which is already loaded. The first running program deletes IERRC9 and starts the processing of records.

SELECTING MODULES FOR THE FINAL MERGE PHASE

Module IERRC8 determines the characteristics of the sorting or merging application by checking PPISW1 in the phase-to-phase information area and sets the corresponding bits in WSWITCH. IERRC8 ORs WSWITCH into each entry in TBLPH3RN, a table of phase 3 running module masks. When the OR operation yields all ones, the module is required for sort/merge execution. IERRC8 stores the last three characters of each needed module in the phase-to-phase information area and prepares a load list of user modification routines for phase 3.

The phase 3 assignment module masks are in table TBLPH3AS. IERRC8 selects the required assignment modules and places their names in a load list. Referring to the load list and the phase-to-phase information area, module IERRC9 loads the required modules.

If messages are to be printed in the final merge phase, requests are made by the running and assignment modules and the actual printing is done by the control module. If this phase is terminated for any condition, control is returned to the phase control module, which, in turn, returns control to the sort system interface module IERRCB.

The assignment modules initialize the final merge phase. Their functions include:

1. Setting up the buffer areas for the merge modules.

2. Generating DCBs and IOBs.

3. Opening the input data sets for a sorting application.

4. Opening the input data sets for a merging application.

5. Modifying the running-program modules to adapt them to a given application.

The actual processing of records is performed by the running-program modules. These modules open the output data sets, read the sequences into main storage, merge them into one final sequence, and then write the combined sequence on the indicated output device.

The main functions of running modules in the final merge phase are as follows:

1. Determine if this is a final merge or merge only operation.
2. Initiate input operations.
3. Open the output data set.
4. Deblock records and pass them to the merge network.
5. Sequence records using binary-insertion technique.
6. Block records when received from the merge network.
7. Close files on detecting the EOF.
8. Check record totals.

After the final merge phase has produced a single ordered record sequence, control is returned to IERRCV to terminate the job.

For a merging application, record input and output operations are performed by the control program's data management routines. For the final merge of a sorting application, the sort/merge program handles the record input operation, and data management is used to handle the record output operation, as in a merging case. (Refer to "Record Movement in the Sort/Merge Program.") The record sequences appearing on the final merge phase input devices are combined into one final sequence in a single merge pass.

A set of decision tables (Charts 72 and 73) indicates the final merge phase modules used for a given application.

Sort/Merge Program Flowcharts and Tables

This series of flowcharts describes the overall organization of each individual phase and the flow of control from one module to another module. Modules are represented by subroutine blocks in these flowcharts. The names used to identify these blocks are the same as the module names used in the program listing. Where a number appears in a block, it indicates a reference to another flowchart in the section.

Each module name consists of six characters; the first three are always IER. For the reader's convenience, the module names in the flowcharts are mentioned by their last three characters only.

Chart 10. Overall Control of Flow in the Sort/Merge Program

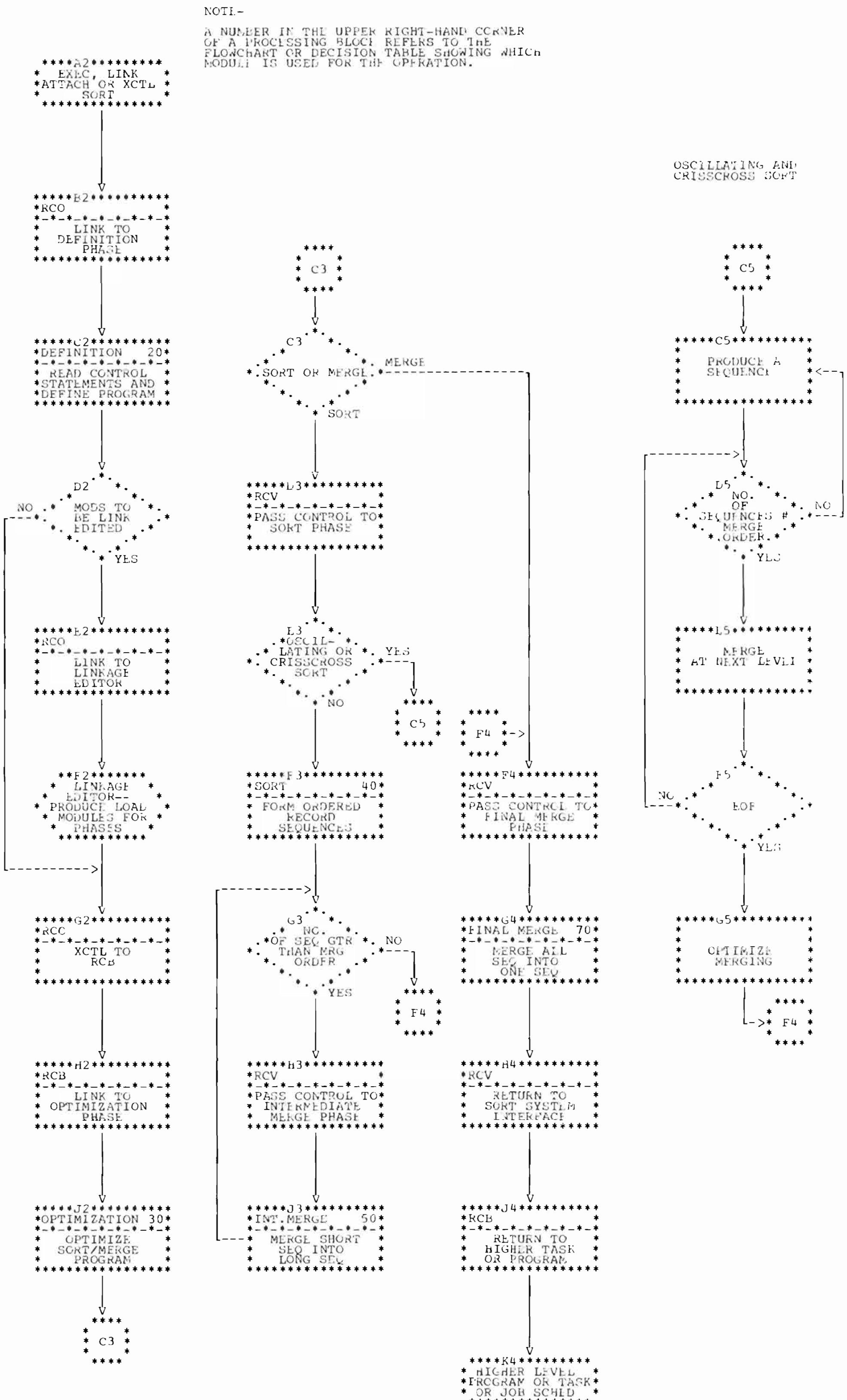
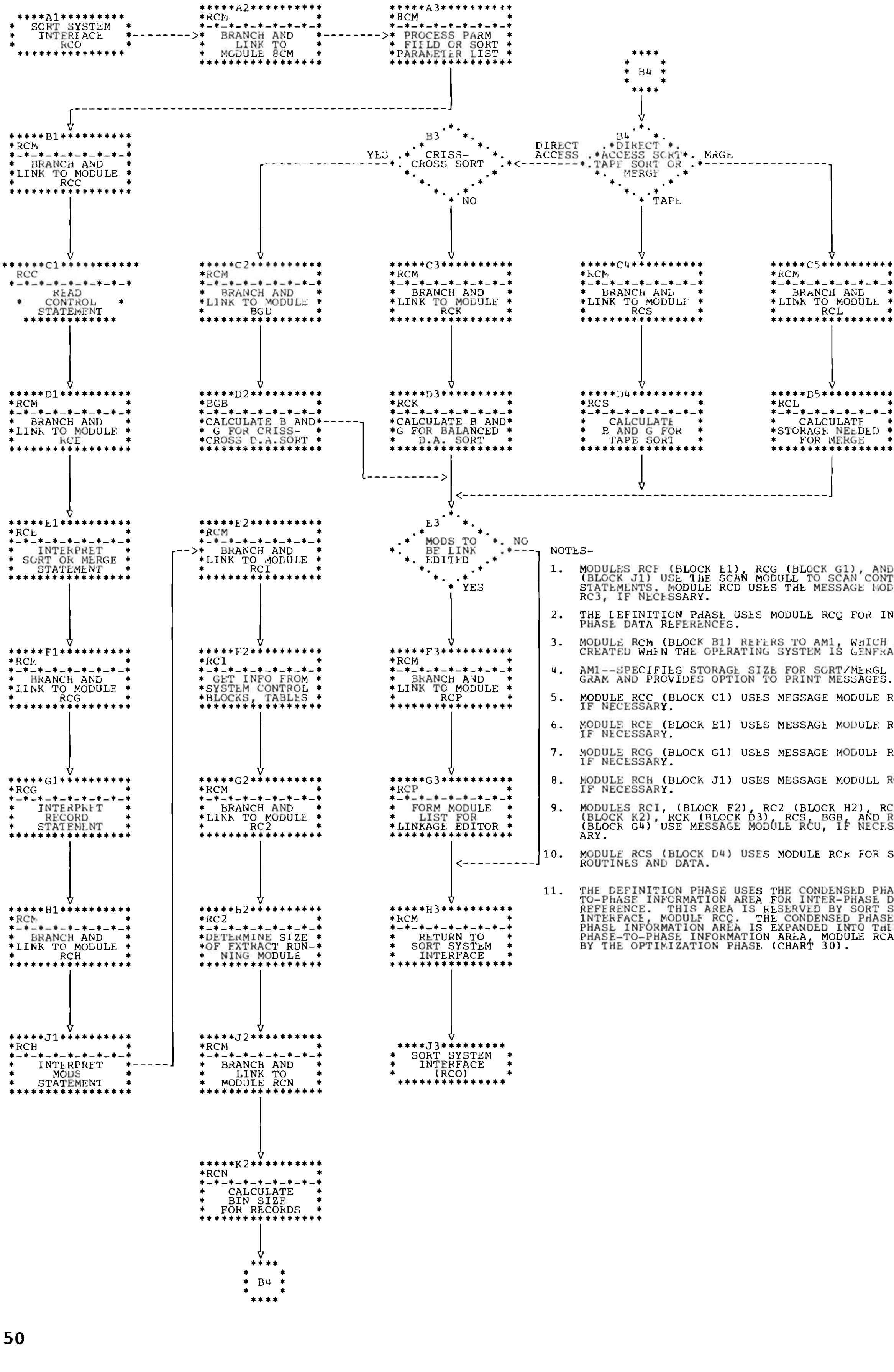


Chart 20. Overall Organization Definition Phase



NOTES-

1. MODULES RCE (BLOCK E1), RCG (BLOCK G1), AND RCH (BLOCK J1) USE THE SCAN MODULE TO SCAN CONTROL STATEMENTS. MODULE RCD USES THE MESSAGE MODULE, RC3, IF NECESSARY.
2. THE DEFINITION PHASE USES MODULE RCQ FOR INTRA-PHASE DATA REFERENCES.
3. MODULE RCM (BLOCK B1) REFERS TO AM1, WHICH IS CREATED WHEN THE OPERATING SYSTEM IS GENERATED.
4. AM1--SPECIFIES STORAGE SIZE FOR SORT/MERGE PRO-GRAM AND PROVIDES OPTION TO PRINT MESSAGES.
5. MODULE RCC (BLOCK C1) USES MESSAGE MODULE RCF, IF NECESSARY.
6. MODULE RCE (BLOCK E1) USES MESSAGE MODULE RCF, IF NECESSARY.
7. MODULE RCG (BLOCK G1) USES MESSAGE MODULE RCW, IF NECESSARY.
8. MODULE RCH (BLOCK J1) USES MESSAGE MODULE RCX, IF NECESSARY.
9. MODULES RCI, (BLOCK F2), RC2 (BLOCK H2), RCN (BLOCK K2), RCK (BLOCK D3), RCS, BGB, AND RCP (BLOCK G4) USE MESSAGE MODULE RCU, IF NECESSA-ARY.
10. MODULE RCS (BLOCK D4) USES MODULE RCR FOR SUB-ROUTINES AND DATA.
11. THE DEFINITION PHASE USES THE CONDENSED PHASE-TO-PHASE INFORMATION AREA FOR INTER-PHASE DATA REFERENCE. THIS AREA IS RESERVED BY SORT SYSTEM INTERFACE, MODULE RCQ. THE CONDENSED PHASE-TO-PHASE INFORMATION AREA IS EXPANDED INTO THE PHASE-TO-PHASE INFORMATION AREA, MODULE RCA, BY THE OPTIMIZATION PHASE (CHART 30).

Chart 30. Overall Organization Optimization Phase

NOTES-

1. THE OPTIMIZATION PHASE USES THE PHASE-TO-PHASE INFORMATION AREA FOR INTER-PHASE DATA REFERENCE.
2. MODULE RC1 TRANSFERS INFORMATION FROM THE CONDENSED PHASE-TO-PHASE INFORMATION AREA INTO THE PHASE-TO-PHASE INFORMATION AREA, MODULE RCA. THE CONDENSED PHASE-TO-PHASE INFORMATION AREA IS RESERVED BY SORT SYSTEM INTERFACE, MODULE RC0, AND FILLED BY THE DEFINITION PHASE (CHART 20).
3. THE OPTIMIZATION PHASE USES MESSAGES MODULE RCU, IF NECESSARY.

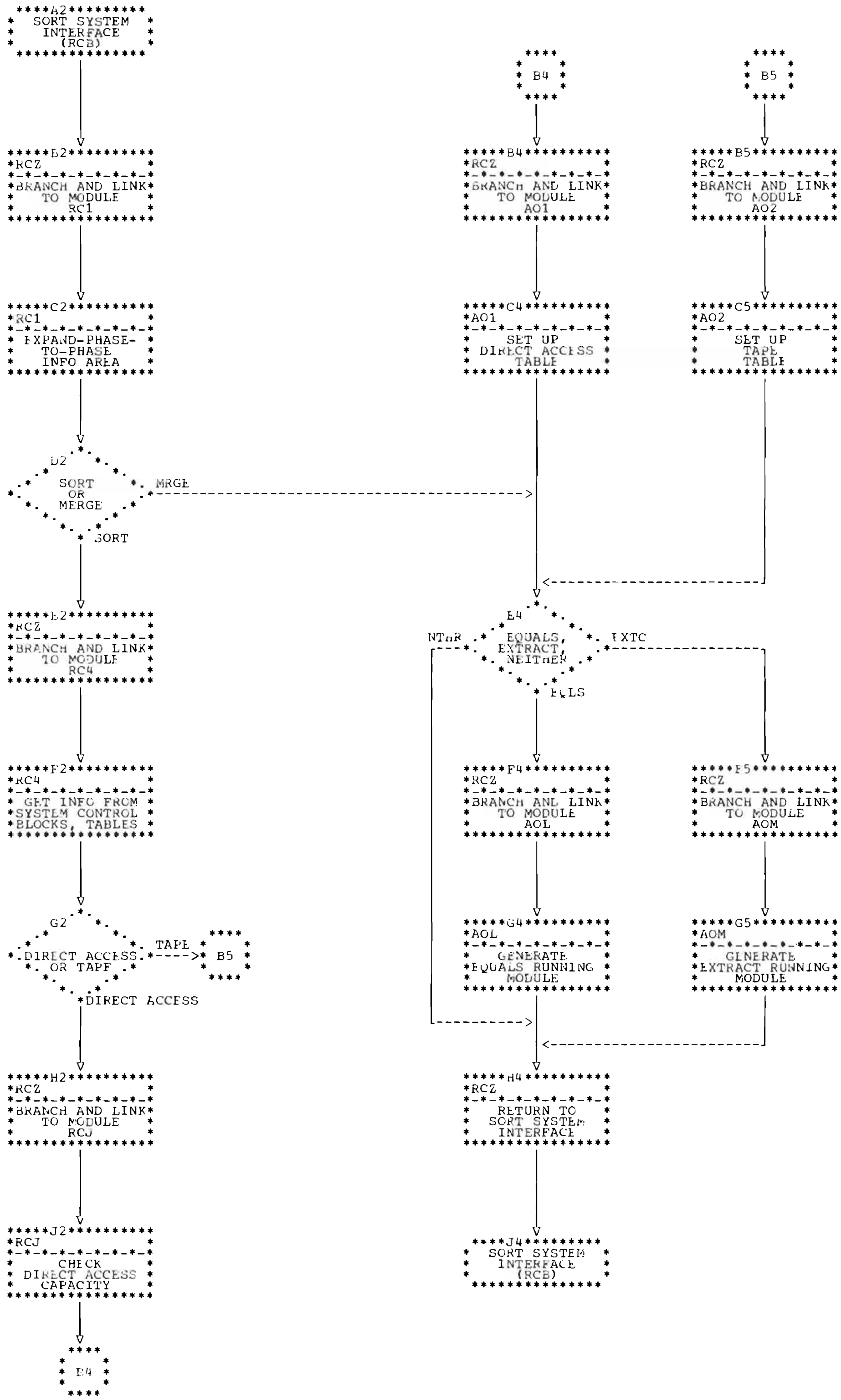


Chart 40. Overall Organization of Sort Phase for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE FLOWCHART OR DECISION TABLE SHOWING WHICH MODULE IS USED FOR THE OPERATION.
2. THE SORT PHASE USES THE PHASE-TO-PHASE INFORMATION AREA, MODULE RCA, FOR DATA REFERENCE.
3. THE SORT PHASE USES MESSAGE MODULE RMA, IF NECESSARY.
4. THE SORT MODULE (BLOCK F3) USES THE EQUALS MODULE IF RECORDS HAVE MORE THAN ONE CONTROL DATA FIELD AND EXTRACT IS NOT USED.
5. THE SORT MODULE (BLOCK F3) USES THE EXTRACT MODULE (WITH EXIT E61) IF USER MODIFICATIONS TO RECORDS ARE MADE BEFORE SORTING, OR IF RECORDS CONTAIN OTHER THAN-
   (1) LOGICAL DATA ON BYTE BOUNDARIES, OR
   (2) CHARACTER DATA
6. THE SEQUENCE DISTRIBUTION MODULE FOR BALANCED DIRECT ACCESS (BLOCK J4) USES THE WRITE MODULE AS A SUBROUTINE.
7. RCV IS GIVEN CONTROL BY RCB. RCV KEEPS CONTROL UNTIL THE TASK IS COMPLETED. IT THEN RETURNS TO RCB WHICH GIVES CONTROL TO THE NEXT TASK.
8. USER EXITS-- E15 FROM THE DEBLOCK MODULE (BLOCK F2)
   E16 FROM THE DEBLOCK MODULE (BLOCK F2)
   E17 FROM MODULE RPC, WHICH IS CALLED BY MODULE RCV (BLOCK H5)
   E61 FROM SORT MODULE (BLOCK F3)

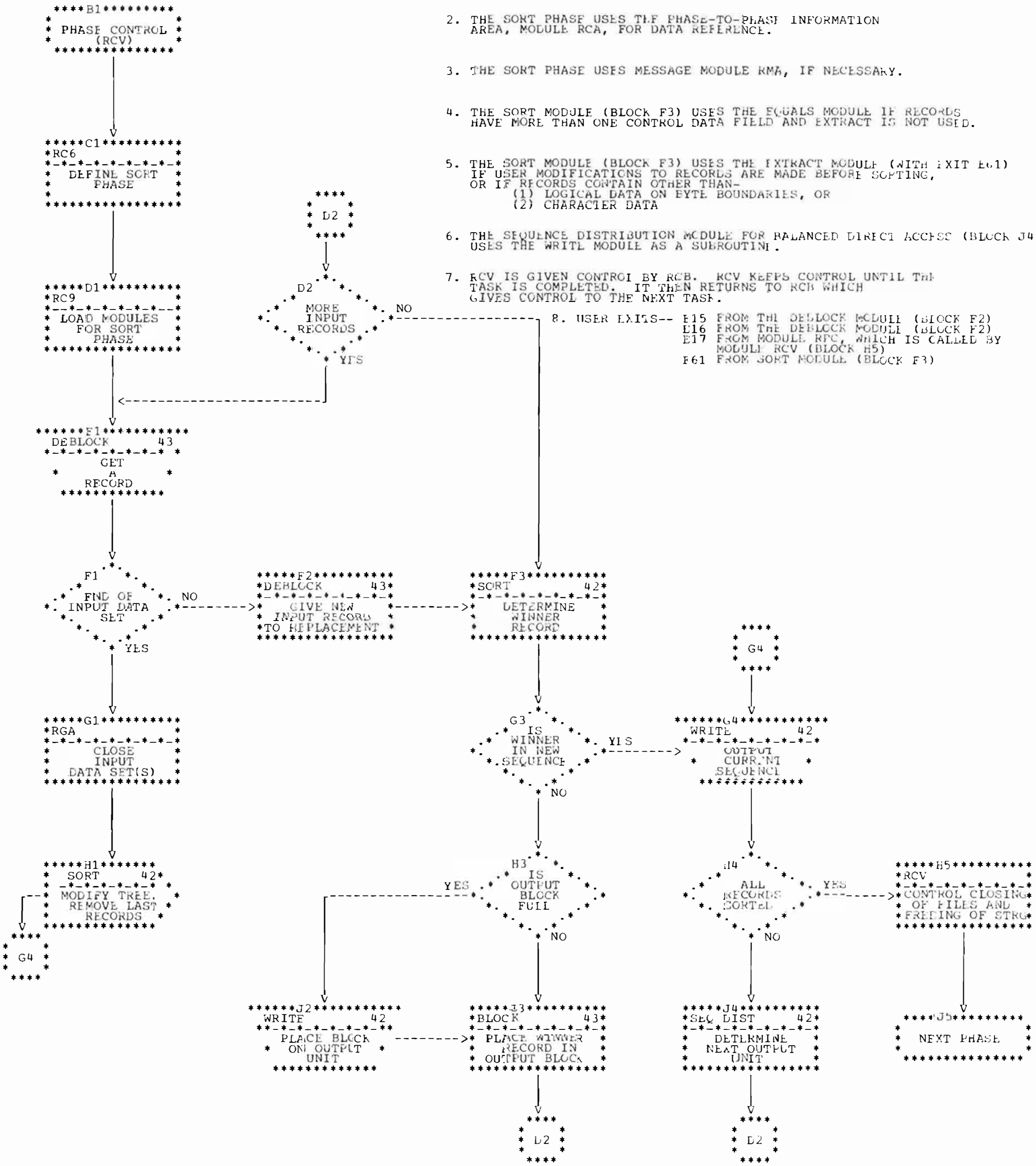


Chart 41. Sort Phase Assignment for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE FLOWCHART OR DECISION TABLE SHOWING WHICH MODULE IS USED FOR THE OPERATION.

2. MODULE AMA IS AN AREA SET ASIDE FOR SORT PHASE MESSAGES. THIS AREA IS ESTABLISHED BEFORE EXECUTION OF MODULE APG.

3. THE DCB GENERATION MODULES (BLOCK K2) USE THE PARAMETER AREA, AP1, TO CONTAIN THE ADDRESSES OF GENERATED DCBS AND DCB OPTIONS.

4. THE OPEN MODULE (BLOCK G5) AND THE WRITE MODULE (BLOCK G3) REFER TO THE PARAMETER AREA, AP1.

5. AN INCOMPLETE LINE BETWEEN MODULE BLOCKS INDICATES THAT EACH TIME AN ASSIGNMENT PROGRAM IS LOADED AND EXECUTED, CONTROL RETURNS TO MODULE RC9.

6. MODULE APC (BLOCK G5) BRANCHES AND LINKS TO MODULE CHK, WHICH ISSUES CHECKPOINT MACRO INSTRUCTIONS, IF REQUESTED.

7. USER EXITS -- E11 FROM MODULE APC (BLOCK D2)
   E18 FROM DCB, IOB GENERATOR (BLOCK K2)
   E19 FROM DCB, IOB GENERATOR (BLOCK K2)

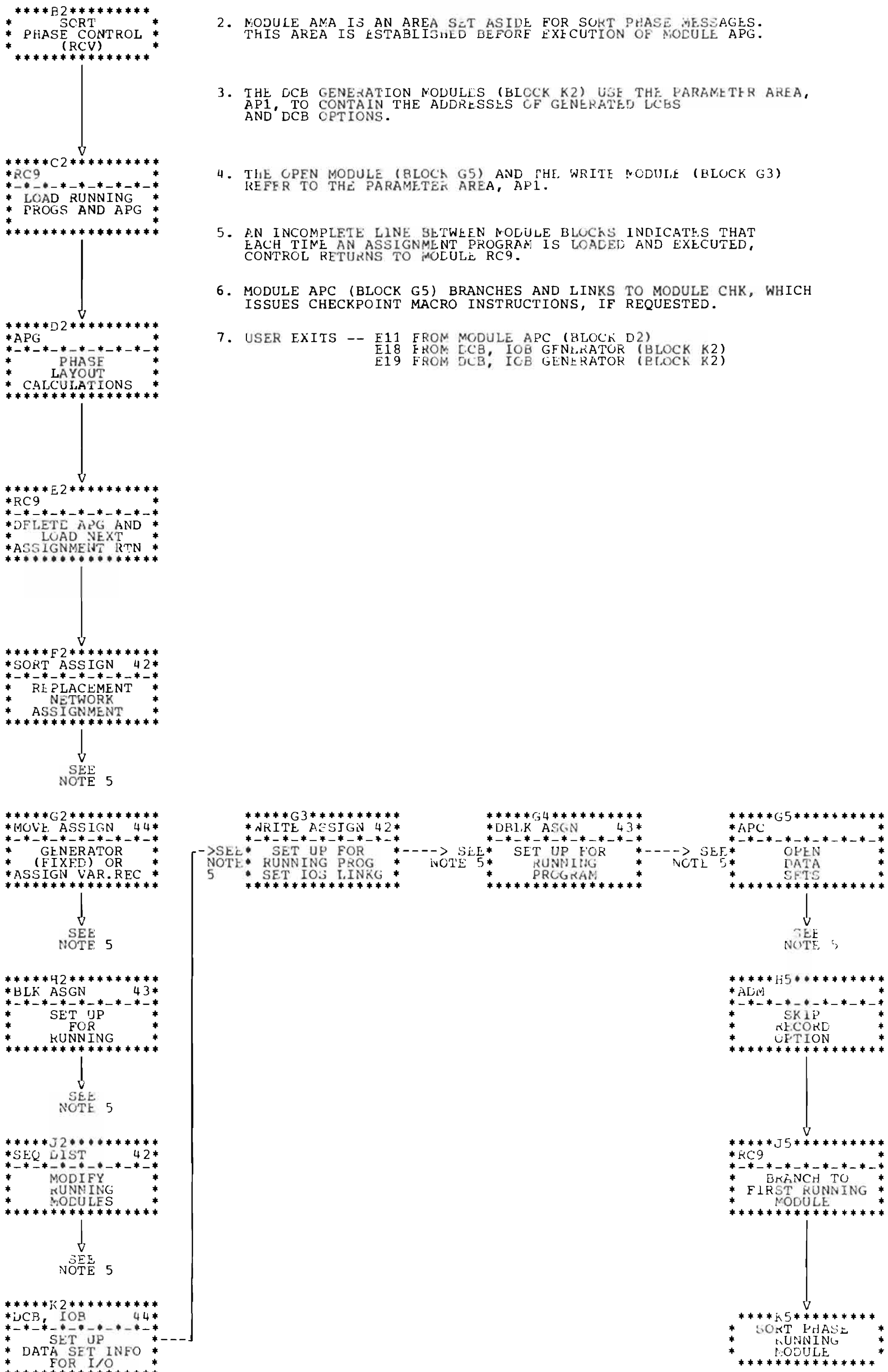


Chart 42. Sort Phase Decision Tables

SORTING

Use	For techniques other than Polyphase Tape	Polyphase Tape	Fixed-length Records	Variable-length Records	Single control fields or extracts used	Multiple control fields. Equals used
Module AOA ROA	X		X			X
AOB ROB	X		X		X	
AOC ROC	X			X		X
AOD ROD	X			X	X	
AOE ROE		X	X			X
AOF ROF		X	X		X	
AOG ROG		X		X		X
AOH ROH		X		X	X	

SEQUENCE DISTRIBUTION

Use	Balanced			Poly-phase	Oscil-lating	Criss-cross
	Tape	Disk	Drum	Tape	Tape	(2314)
Module AOI ROI	X					
AOJ ROJ				X		
AOK ROK		X				
AOO ROO			X			
AON RON					X	
8ON 9ON						X

WRITING

Use		Balanced		Criss-cross
	Tape	Disk	Drum	(2314)
Module APA RPA	X			
APB RPB		X		
APN RPN			X	
8PA 9PA				X

Chart 43. Sort Phase Decision Tables

DEBLOCK - BALANCED TAPE, BALANCED DIRECT ACCESS, OR POLYPHASE TAPE TECHNIQUE

Use	Fixed-length Records	Variable-length Records	Fixed-length In-line move <256	Fixed-length Multiple move >256	Variable Move	User Exits	No User Exits
<u>Module</u> ADB RDB	X		X				X
ADC RDC	X			X			X
ADD RDD	X			X		E15 E16	
ADE RDE		X			X	E15 E16	
ADG RDG		X			X		X

BLOCK - BALANCED TAPE, BALANCED DIRECT ACCESS, OR POLYPHASE TAPE TECHNIQUE

ABB RBB	X		X				X
ABC RBC	X			X			X
ABE RBE		X			X		X

Chart 44. Sort Phase Decision Tables

DEBLOCK - OSCILLATING OR CRISSCROSS TECHNIQUE

Use	Fixed-length Records	Variable-length Records	Fixed-length In-line move <256	Fixed-length Multiple move <256	Variable Move	User Exits	No User Exits
Module ADP RDP	X		X				X
ADQ RDQ	X			X			X
ADR RDR	X					E15 E16	
ADS RDS		X			X	E15 E16	
ADT RDT		X			X		X

BLOCK - OSCILLATING OR CRISSCROSS TECHNIQUE

Use	Fixed-length Records	Variable-length Records	Fixed-length In-line move <256	Fixed-length Multiple move <256	Variable Move	User Exits	No User Exits
ABA RBA		X			X		X
ABY RBY	X		X				X
ABZ RBZ	X			X			X

MOVE

Use	Fixed-length Records	Variable-length Records
Module ABF RBF		X
ABS	X	

GENERATE DCB, IOB

Use	Tape	Balanced Disk/ Drum	Oscillating Tape	Criss-cross (2314)
Module AGA	X			
AGI		X		
AGN			X	
9GN				X

Chart 50. Overall Organization of Intermediate Merge Phase for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE FLOWCHART OR DECISION TABLE SHOWING WHICH MODULE IS USED FOR THE OPERATION.
2. THE INTERMEDIATE MERGE PHASE USES THE PHASE-TO-PHASE INFORMATION AREA, MODULE RCA, FOR DATA REFERENCE.
3. THE INTERMEDIATE MERGE PHASE USES MESSAGE MODULE RMB, IF NECESSARY.
4. THE MERGE MODULE (BLOCKS G1 AND E3) USES THE EQUALS MODULE IF MORE THAN ONE CONTROL DATA FIELD IS PRESENT PER RECORD AND IF EXTRACT MODULE IS NOT USED.
5. THE MERGE MODULE (BLOCKS G1 AND E3) USES THE EXTRACT MODULE (WITH EXIT E61) IF USER MODIFICATIONS ARE MADE OR IF RECORDS CONTAIN OTHER THAN-
   (1) LOGICAL DATA ON BYTE BOUNDARIES, OR
   (2) CHARACTER DATA.
6. THE BALANCED DIRECT ACCESS SEQUENCE DISTRIBUTION MODULE (BLOCK G5) USES BOTH THE READ MODULE AND THE WRITE MODULE AS SUBROUTINES.
7. USER EXITS -- E25 FROM BLOCK MODULE (BLOCK G2)
   E27 FROM MODULE RPF, WHICH IS CALLED BY MODULE RCV (BLOCK E3)
   E61 FROM MERGE MODULE (BLOCK E3)
8. THE SEQ DIST MODULE (BLOCK G5) BRANCHES AT CERTAIN INTERVALS TO MODULE CHK, WHICH ISSUES CHECKPOINT MACRO INSTRUCTIONS, IF REQUESTED.

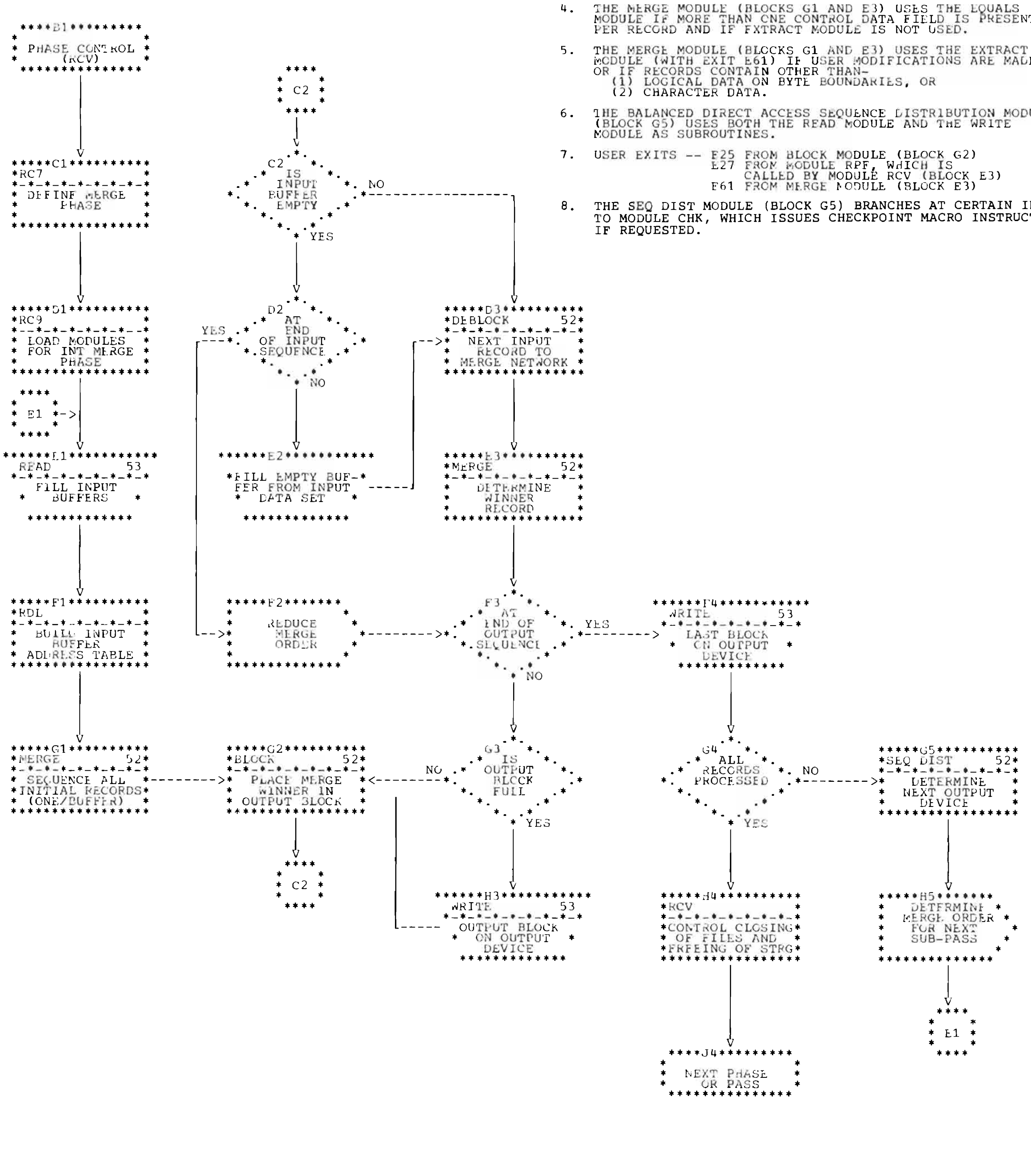


Chart 51. Intermediate Merge Phase Assignment for Balanced Direct Access, Polyphase Tape, and Balanced Tape Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE DECISION TABLE SHOWING WHICH MODULE WILL BE USED FOR THE OPERATION.
2. MODULE AMB IS AN AREA SET ASIDE FOR INTERMEDIATE MERGE PHASE MESSAGES. THIS AREA IS ESTABLISHED BEFORE EXECUTION OF MODULE APH.
3. THE DCB GENERATION MODULES (BLOCK F3) USE THE PARAMETER AREA, AP2, TO CONTAIN ADDRESSES OF GENERATED DCBS AND DCB OPTIONS.
4. THE OPEN MODULE (BLOCK G5), THE WRITE MODULE (BLOCK F4), AND THE READ MODULE (BLOCK H5) REFER TO THE PARAMETER AREA, AP2.
5. AN INCOMPLETE LINE BETWEEN MODULE BLOCKS INDICATES THAT EACH TIME AN ASSIGNMENT PROGRAM IS LOADED AND EXECUTED, CONTROL RETURNS TO MODULE RC9.
6. IF CHECKPOINTS ARE REQUESTED, MODULE APJ (BLOCK G5) WILL BRANCH TO MODULE CHK, WHICH ISSUES CHECKPOINT MACRO INSTRUCTIONS.
7. USER EXITS -- E21 FROM MODULE APH (BLOCK D2)
   E28 FROM DCB, IOB GENERATOR (BLOCK F3)
   E29 FROM DCB, IOB GENERATOR (BLOCK F3)

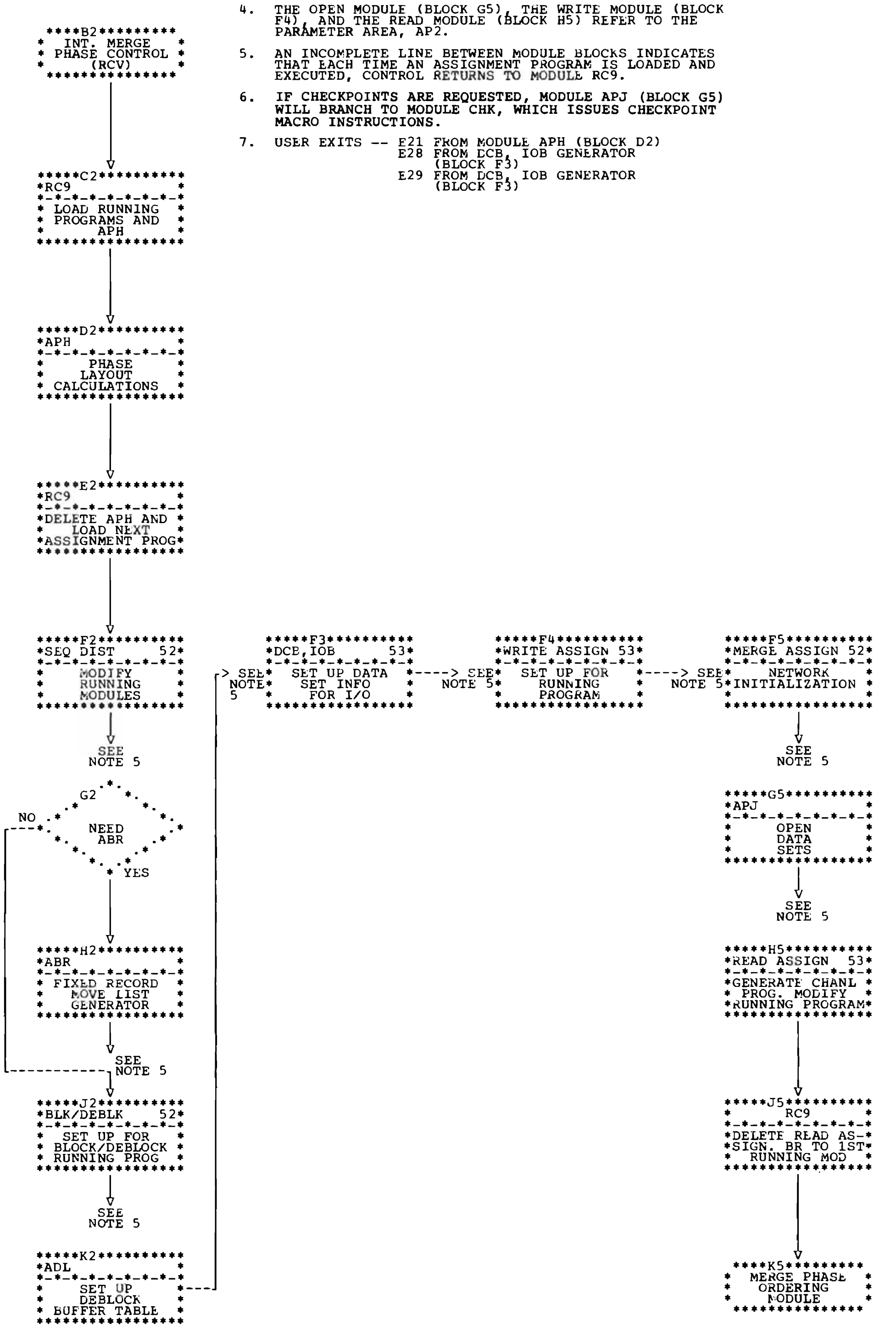


Chart 52. Intermediate Merge Phase Decision Tables

MERGING

Use	Single Control/ Extracts Used	Multiple Control (Equals)	8-way	16-way
<u>Module</u>				
AOP ROP		X		X
AOQ ROQ	X			X
AOU ROU		X	X	
AOV ROV	X		X	

SEQUENCE DISTRIBUTION

Use	Balanced			Poly- phase	Oscil- lating	Criss- cross
	Tape	Disk	Drum	Tape	Tape	(2314)
<u>Module</u>						
AOR ROR	X					
AOS ROS				X		
AOT ROT		X				
AO3 RO3			X			
AON RON					X	
8ON 9ON						X

BLOCK/DEBLOCK - BALANCED TAPE, BALANCED DIRECT ACCESS OR POLYPHASE TAPE TECHNIQUE

Use	Fixed-Length Records	Variable-Length Records	Fixed-Length In-Line move <256	Fixed-Length Multiple move >256	Variable Move	User Exits Indicated	No User Exits
<u>Module</u>							
ABG RBG	X		X				X
ABH RBH	X			X			X
ABI RBI		X			X		X
ABJ RBJ	X			X		E25	
ABK RBK		X			X	E25	

BLOCK/DEBLOCK - OSCILLATING OR CRISSCROSS TECHNIQUE

ABT RBT	X		X				X
ABU RBU	X			X			X
ABV RBV		X			X		X
ABW RBW	X			X		E25	
ABX RBX		X			X	E25	

Chart 53. Intermediate Merge Phase Decision Tables

READ

Use / Module	Backward Tape	Balanced Disk	Balanced Drum	Forward Tape	Oscillating Tape	Crisscross (2314)
AGB RGB	X				X	
AGL RGL				X		
AGC RGC		X				
AGO RGO			X			
8GB 9GB						X

WRITE

Use / Module	Tape	Balanced Disk	Balanced Drum	Crisscross (2314)
APD RPD	X			
APE RPE		X		
APO RPO			X	
8PA 9PA				X

GENERATE DCB, IOB

Use / Module	Tape	Balanced Disk/Drum	Oscillating Tape	Crisscross (2314)
AGG	X			
AGJ		X		
AGN			X	
9GN				X

Chart 60. Overall Organization of Sort/Merge Phases for Oscillating Tape and Crisscross Direct Access Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE DECISION TABLE WHICH IS USED FOR THE OPERATION.
2. USER EXITS -- E15 DROM DEBLOCK MODULE (BLOCK D1)
   E16 FROM DEBLOCK MODULE (BLOCK D1)
   E17 FROM EITHER MODULE RPM OR 8PM, WHICH IS CALLED BY MODULE RCV (BLOCK G5)
   E25 FROM BLOCK MODULE (BLOCK G3)
3. IN AN OSCILLATING SORT, IF CHECKPOINT HAS BEEN SPECIFIED, THE SEQ DIST MODULE (BLOCK F4) WILL BRANCH TO MODULE CHK AT CERTAIN INTERVALS. CHK ISSUES CHECKPOINT MACRO INSTRUCTIONS.

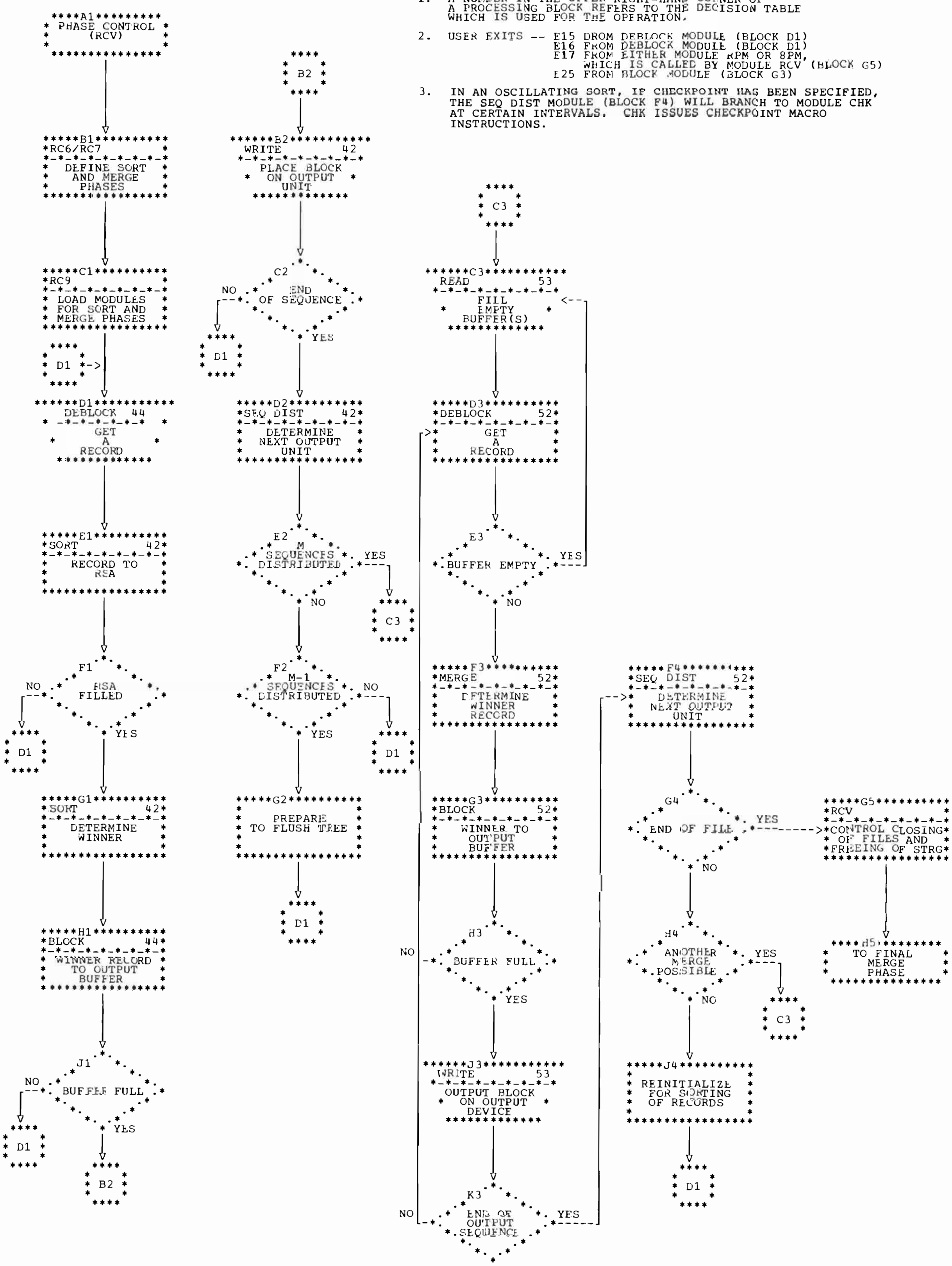


Chart 61. Sort/Merge Phase Assignment for Oscillating Tape and Crisscross Direct Access Techniques

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE DECISION TABLE SHOWING WHICH MODULE IS USED FOR THE OPERATION.
2. AN INCOMPLETE LINE BETWEEN MODULE BLOCKS INDICATES THAT EACH TIME AN ASSIGNMENT PROGRAM IS LOADED AND EXECUTED, CONTROL RETURNS TO MODULE RC9.
3. MODULE APC (BLOCK G4) BRANCHES TO MODULE CHK, WHICH ISSUES CHECKPOINT MACRO INSTRUCTIONS, IF REQUESTED.
4. USER EXITS -- E11 FROM MODULE APL (BLOCK D1)
   E18 FROM DCB, IOB GENERATOR (BLOCK D2)
   E19 FROM DCB, IOB GENERATOR (BLOCK D2)
   E21 FROM MODULE APL (BLOCK D1)
   E28 FROM DCB, IOB GENERATOR (BLOCK D2)
   E29 FROM DCB, IOB GENERATOR (BLOCK D2)

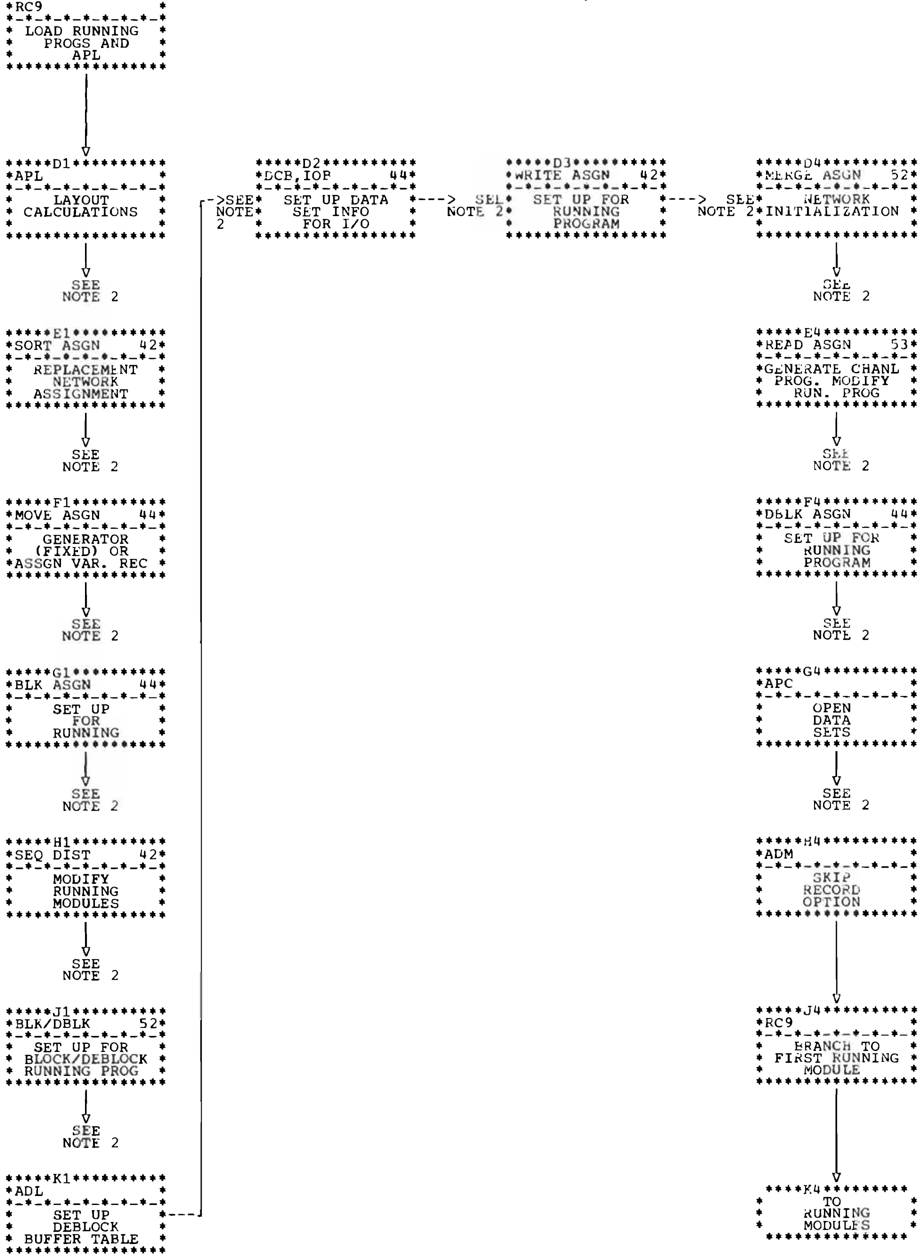


Chart 70. Overall Organization of Final Merge Phase

NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE DECISION TABLE SHOWING WHICH MODULE IS USED FOR THE OPERATION.

2. THE FINAL MERGE PHASE USES THE PHASE-TO-PHASE INFORMATION AREA, MODULE RCA, FOR DATA REFERENCES.

3. THE MERGE PHASE USES MESSAGE MODULE RMC, IF NECESSARY.

4. THE MERGE MODULE (BLOCKS E1 AND E3) USES THE EQUALS MODULE IF MORE THAN ONE CONTROL DATA FIELD IS PRESENT PER RECORD AND IF EXTRACT MODULE IS NOT USED.

5. THE MERGE MODULE (BLOCKS E1 AND E3) USES THE EXTRACT MODULE (WITH EXIT E61) IF USER MODIFICATIONS ARE MADE, OR IF RECORDS CONTAIN OTHER THAN-
   (1) LOGICAL DATA ON BYTE BOUNDARIES, OR
   (2) CHARACTER DATA

6. FOR MERGING APPLICATIONS, THE DEBLOCK MODULE (BLOCK D3) ISSUES A 'GET' TO REFILL BUFFER AREAS.

7. USER EXITS -- E35 FROM BLOCK MODULE (BLOCK F4)
   E37 FROM MODULE RPG, WHICH IS CALLED BY MODULE RCV (BLOCK J4)
   E61 FROM MERGE MODULE (BLOCK E3)

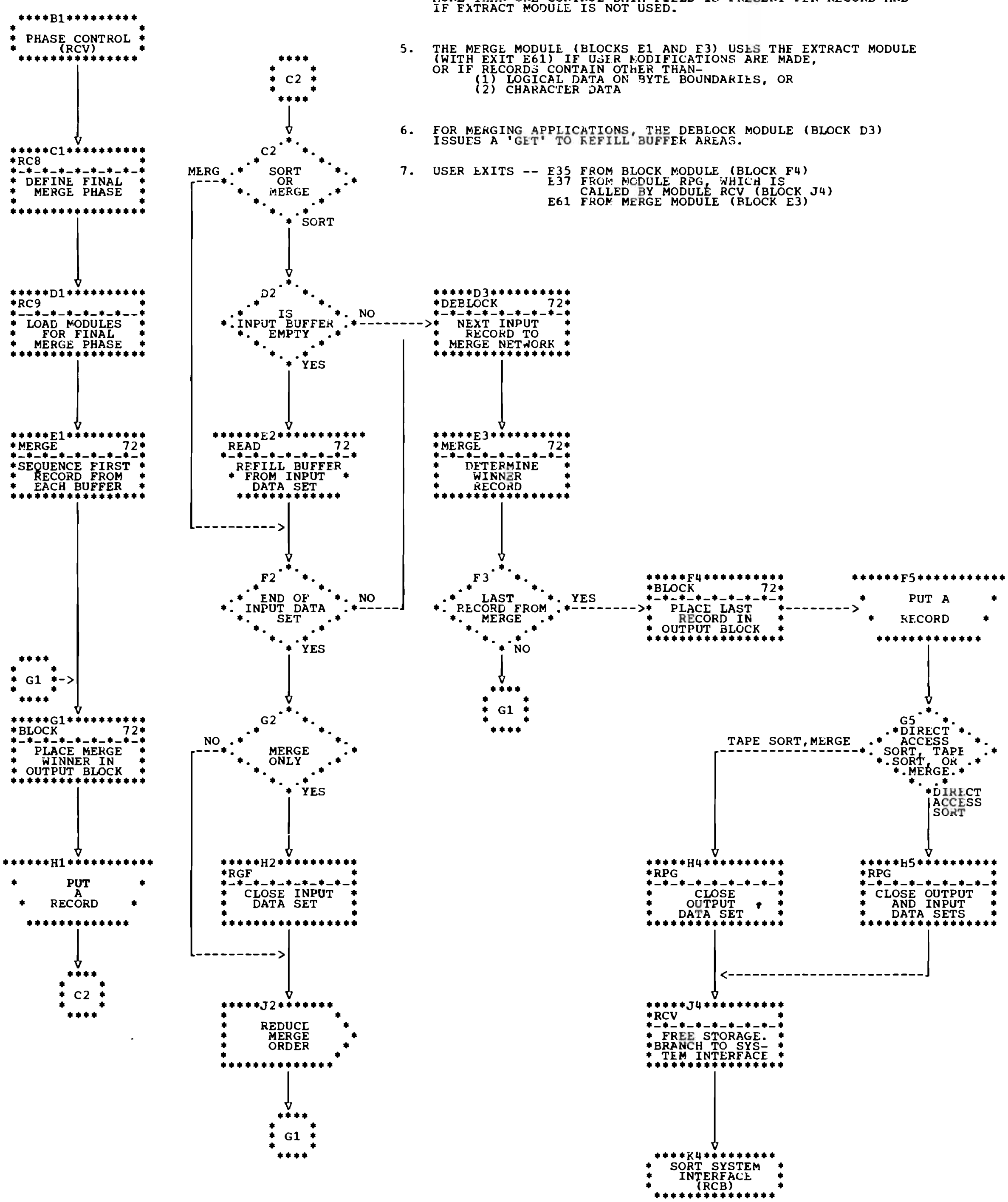
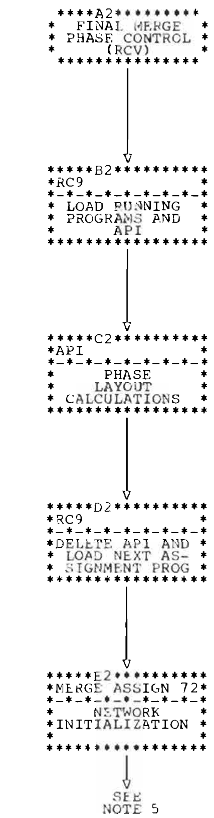


Chart 71. Final Merge Phase Assignment



NOTES-

1. A NUMBER IN THE UPPER RIGHT-HAND CORNER OF A PROCESSING BLOCK REFERS TO THE DECISION TABLE SHOWING WHICH MODULE WILL BE USED FOR THE OPERATION.

2. THE MODULE AMC IS AN AREA SET ASIDE FOR FINAL MERGE PHASE MESSAGES. THIS AREA IS ESTABLISHED BEFORE EXECUTION OF MODULE API.

3. THE DCB GENERATION MODULES (BLOCK F2) USES THE PARAMETER AREA, AP3, TO CONTAIN ADDRESSES OF DCBS AND DCB OPTIONS.

4. THE OPEN MODULE (BLOCK F4) AND THE READ MODULE (BLOCK G4) REFER TO THE PARAMETER AREA, AP3.

5. AN INCOMPLETE LINE BETWEEN MODULE BLOCKS INDICATES THAT EACH TIME AN ASSIGNMENT MODULE IS LOADED AND EXECUTED, CONTROL RETURNS TO MODULE RC9.

6. MODULE AGH (BLOCK F4) BRANCHES TO MODULE CHK, WHICH ISSUES CHECKPOINT MACRO INSTRUCTIONS, IF REQUESTED.

7. USER EXITS -- E31 FROM MODULE API (BLOCK C2)
E38 FROM DCB, IOB GENERATOR (BLOCK F2)
E39 FROM DCB, IOB GENERATOR (BLOCK F2)

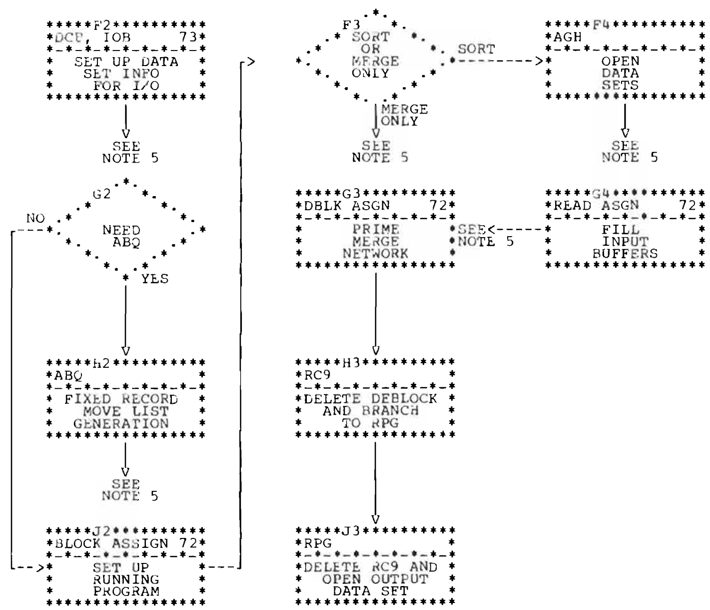


Chart 72. Final Merge Phase Decision Tables

MERGING

Use	Single Control Extracts	Multiple Control (Equals)	8-way	16-way
Module AOP ROP		X		X
AOQ ROQ	X			X
AOU ROU		X	X	
AOV ROV	X		X	

READING

Use	Backward Tape	Balanced		Forward Tape	Criss-cross (2314)
		Disk	Drum		
Module AGD RGD	X				
AGM RGM				X	
AGE RGE		X			
AGP RGP			X		
8GC 9GC					X

DEBLOCK

Use	Fixed-length Records	Variable-length Records		Fixed Move <256	Fixed Move >256	Variable Move	User Exits	No User Exits	Sort	Merge	Forward Tape
		Unspanned	Spanned								
Module ADH RDH	X							X	X		
ADI RDI		X	X					X	X		
ADJ RDJ	X	X						X		X	
ADX RDX	X							X	X		X
8DJ 9DJ			X					X		X	

BLOCK

Use	Fixed-length Records	Unspanned	Spanned	Fixed Move <256	Fixed Move >256	Variable Move	User Exits	No User Exits
ABL RBL	X			X				
ABM RBM	X				X		E35	
ABN RBN		X				X		X
ABO RBO		X				X	E35	
ABP RBP	X				X			X
8BN 9BN			X			X		X
8BO 9BO			X			X	E35	

Chart 73. Final Merge Phase Decision Tables

GENERATE DCB, IOB

Use / Module	Tape	Direct Access	Sort	Merge
AGK		X	X	
APF	X	X		X
APK	X		X	

OPEN DATA SETS

Use / Module	Chkpt	Build	Sort	Merge
AGH	X		X	
AGF		X		X

Section 4: Module Directory

This section contains an alphabetic list of modules in the sort/merge program.

Each sort/merge module name consists of six characters, the first three of which are always IER. For easy reference, the module names are listed by their last three characters only.

CSECT Name	Phase in Which Used	Purpose for Which Used
8BN	Fin.Merge	Block variable-length spanned records
8BO	Fin.Merge	Block variable-length spanned records with modifications
8CI	Def.	Contains tables and constants for module RCI
8CM	Def.	Process PARM field and sort parameter list, initialize module CPI, open SYSOUT if requested
8DJ	Fin.Merge	Deblock variable-length spanned records for a merge operation
8GB	Int.Merge	Read for crisscross technique
8GC	Fin.Merge	Read for crisscross technique
8ON	C-C Sort	Crisscross technique algorithm
8PA	Sort and Int.Merge	Write for crisscross technique
8PM	C-C Sort	End of phase housekeeping for crisscross technique
9BN	Fin.Merge	Assignment block variable-length spanned records
9BO	Fin.Merge	Assignment block variable-length spanned records with modification
9DJ	Fin.Merge	Assignment deblock variable-length spanned records for a merge operation
9GB	Int.Merge	Read assignment for crisscross technique
9GC	Fin.Merge	Read assignment for crisscross technique
9GN	C-C Sort	Generates DCBs, IOBs, ECBs, and alternate CCW pointers
9ON	C-C Sort	Assignment crisscross algorithm
9PA	Sort and Int.Merge	Write assignment for crisscross technique
ABA	Osc. and C-C Sort	Assignment routine sort block variable-length records
ABB	Sort	Block fixed-length records with in-line move
ABC	Sort	Block fixed-length records with link to multiple move
ABE	Sort	Block variable-length records with move
ABF	Sort	Move module for variable-length records
ABG	Int.Merge	Block or deblock fixed-length records with in-line move
ABH	Int.Merge	Block or deblock fixed-length records with link to multiple move
ABI	Int. Merge	Block/deblock variable-length records - multiple move
ABJ	Int.Merge	Block or deblock fixed-length records with modification
ABK	Int. Merge	Block/deblock variable-length records
ABL	Fin.Merge	Block fixed-length records with in-line move
ABM	Fin.Merge	Block fixed-length records with modifications

(Part 1 of 6)

CSECT Name	Phase in Which Used	Purpose for Which Used
ABN	Fin.Merge	Block variable-length records
ABO	Fin.Merge	Block variable-length records with modifications
ABP	Fin.Merge	Block fixed-length records with link to multiple move
ABQ	Fin.Merge	Move generator for fixed-length records
ABR	Int.Merge	Move generator for fixed-length records
ABS	Sort	Move generator for fixed-length records
ABT	Osc. and C-C Merge	Assignment routine for merge block/deblock fixed-length records<256 bytes
ABU	Osc. and C-C Merge	Assignment routine for merge block/deblock fixed-length records>256 bytes
ABV	Osc. and C-C Merge	Assignment routine for merge block/deblock variable-length records
ABW	Osc. and C-C Merge	Assignment routine for merge block/deblock fixed-length records with user exits
ABX	Osc. and C-C Merge	Assignment routine for merge block/deblock variable-length records with user exits
ABY	Osc. and C-C Sort	Assignment routine sort block fixed-length records<256 bytes
ABZ	Osc. and C-C Sort	Assignment routine sort block fixed-length records>256 bytes
ADB	Sort	Deblock fixed-length records with in-line move
ADC	Sort	Deblock fixed-length records with link to multiple move
ADD	Sort	Deblock fixed-length records with user exits
ADE	Sort	Deblock variable-length records with user exits
ADG	Sort	Deblock variable-length records
ADH	Fin.Merge	Deblock fixed-length records for a sort
ADI	Fin.Merge	Deblock variable-length records for a sort
ADJ	Fin.Merge	Deblock for a merge
ADL	Int. Merge	Assignment phase builds tables of input buffer address
ADM	Sort	Skip record option
ADP	Osc. and C-C Sort	Assign sort deblock fixed-length records<256 bytes
ADQ	Osc. and C-C Sort	Assign sort deblock fixed-length records>256 bytes
ADR	Osc. and C-C Sort	Assign sort deblock fixed-length records with user exits
ADS	Osc. and C-C Sort	Assign sort deblock variable-length records with user exits
ADT	Osc. and C-C Sort	Assign sort deblock variable-length records without user exits
ADX	Fin. Merge-Sorting appl.	Deblock assignment for read-forward, fixed-length tape sort
AGA	Sort	Generate DCBs, IOBs, and DCB addresses; tape only
AGB	Int.Merge	Read tape assignment
AGC	Int.Merge	Read disk assignment
AGD	Fin.Merge	Read tape assignment
AGE	Fin.Merge	Read disk assignment
AGF	Fin.Merge	Open files
AGG	Int.Merge	Generate DCBs, IOBs, and DCB addresses; tape only
AGH	Fin.Merge	Open files and initiate checkpoint operations
AGI	Sort	Generate DCBs, IOBs, and DCB addresses; disk only
AGJ	Int.Merge	Generate DCBs, IOBs, and DCB addresses; disk only
AGK	Fin.Merge	Generate DCBs, IOBs, and DCB addresses; disk only
AGL	Int. Merge	Read forward assignment - tape sort
AGM	Fin. Merge-Sorting appl.	Read forward assignment - tape sort
AGN	Osc. Sort	Generate DCBs and IOBs
AGO	Int. Merge	Read assignment for drum
AGP	Fin. Merge	Read assignment for drum
AMA	Sort	Contain all messages for the sort phase assignment modules

(Part 2 of 6)

CSECT Name	Phase in Which Used	Purpose for Which Used
AMB	Int.Merge	Contain all messages for the merge phase assignment modules
AMC	Fin.Merge	Contain all messages for the final merge and merge-only phases assignment modules
AM1	Def.	Specify core storage for the sort/merge program and also provide option for printing messages
AOA	Sort	Replace sort control for fixed-length records
AOB	Sort	Replace sort control for fixed-length records
AOC	Sort	Replace sort control for variable-length records
AOD	Sort	Replace sort control for variable-length records
AOE	Sort	Replace sort control for fixed-length records in polyphase
AOF	Sort	Replace sort control for fixed-length records in polyphase
AOG	Sort	Replace sort control for variable-length records in polyphase
AOH	Sort	Replace sort control for variable-length records in polyphase
AOI	Sort	Balanced tape sort algorithm
AOJ	Sort	Polyphase sort algorithm
AOK	Sort	Balanced disk sort algorithm
AOL	Opt.	Equals routine
AOM	Opt.	Extract routine
AON	Osc. Sort	Assignment oscillating algorithm
AOO	Sort	Assignment drum-sort algorithm
AOP	Int. & Fin. Merge	16-way merge network with multiple control fields
AOQ	Int. & Fin. Merge	16-way merge network with single control fields
AOR	Int. Merge	Balanced tape merge algorithm
AOS	Int.Merge	Polyphase tape merge algorithm
AOT	Int.Merge	Balanced disk merge algorithm
AOU	Int. & Fin. Merge	8-way merge network with multiple control fields
AOV	Int. & Fin. Merge	8-way merge network with single control fields
AOW	Osc. and C-C Sort	Assignment - oscillating initialization routine, multiple control fields fixed-length records
AOX	Osc. and C-C Sort	Assignment - oscillating initialization routine, single control fields fixed-length records
AOY	Osc. and C-C Sort	Assignment - oscillating initialization routine, multiple control fields variable-length records
AOZ	Osc. and C-C Sort	Assignment - oscillating initialization routine, single control fields variable-length records
AO1	Opt.	Optimize disk/drum unit assignment
AO2	Opt.	Optimize tape unit assignment
AO3	Int. Merge	Drum merge algorithm
APA	Sort	Write tape assignment
APB	Sort	Write disk assignment
APC	Sort	Open files and initiate checkpoint operations
APD	Int.Merge	Write tape assignment
APE	Int.Merge	Write disk assignment
APF	Fin.Merge	Generate DCBs and DCB addresses
APG	Sort	Calculate storage for I/O buffers, RSA, and generated cores
APH	Int.Merge	Calculate storage for I/O buffers and generated cores
API	Fin.Merge & Merge-only	Calculate storage for I/O buffers and generated cores
APJ	Int.Merge	Open files and initiate checkpoint operations
APK	Fin.Merge	Generate DCBs, IOBs, and DCB addresses; tape only
APL	Osc. and C-C Sort	Calculate storage for I/O buffers, RSA, and generated cores
APN	Sort	Drum write assignment routine
APO	Int. Merge	Drum write assignment routine
AP1	Sort	Specify area for DCB list for open
AP2	Int.Merge	Specify area for DCB list for open
AP3	Fin.Merge	Specify area for DCB list for open

(Part 3 of 6)

CSECT Name	Phase in Which Used	Purpose for Which Used
BGA	Opt.	Calculate B and G values for tape
BGB	Opt.	Calculate B and G values for crisscross application
CHK	All phases	Checkpoint routine
DM4	All phases	Hexadecimal and decimal conversion routine
EX1	Sort	Resolve user-exits entries link edited for sort phase
EX2	Int. Merge	Resolve user-exits entries link edited for Int. merge phase
EX3	Fin. Merge	Resolve user-exits entries link edited for Fin. merge phase
RBA	Osc. and C-C Sort	Block variable-length record
RBB	Sort	Block fixed-length records with in-line move
RBC	Sort	Block fixed-length records with link to multiple move
RBE	Sort	Block variable-length records with move
RBF	Sort	Move module for variable-length records
RBG	Int.Merge	Block or deblock fixed-length records with in-line move
RBH	Int.Merge	Block or deblock fixed-length records with link to multiple move
RBI	Int.Merge	Block or deblock variable-length records with move
RBJ	Int.Merge	Block or deblock fixed-length records with modifications
RBK	Int.Merge	Block or deblock variable-length records with modifications
RBL	Fin.Merge	Block fixed-length records with in-line move
RBM	Fin.Merge	Block fixed-length records with modifications
RBN	Fin.Merge	Block variable-length records
RBO	Fin.Merge	Block variable-length records with modifications
RBP	Fin.Merge	Block fixed-length records with link to multiple move
RBT	Osc. and C-C Merge	Merge block/deblock fixed-length record<256 bytes
RBU	Osc. and C-C Merge	Merge block/deblock fixed-length record>256 bytes
RBV	Osc. and C-C Merge	Merge block/deblock variable-length record
RBW	Osc. and C-C Merge	Merge block/deblock fixed-length record with user exits
RBX	Osc. and C-C Merge	Merge block/deblock variable-length record with user exits
RBY	Osc. and C-C Sort	Sort block fixed-length record<256 bytes
RBZ	Osc. and C-C Sort	Sort block fixed-length record>256 bytes
RCA	All but Def.	Specify phase-to-phase information area
RCB	All but Def.	Sort system interface for the sort/merge program
RCC	Def.	Read control cards
RCD	Def.	Scan control cards
RCE	Def.	Interpret sort or merge cards
RCF	Def.	Contains error messages for sort/merge cards
RCG	Def.	Interpret record cards
RCH	Def.	Interpret modification cards
RCI	Def.	Search for control system
RCJ	Opt.	Check direct access capacity
RCK	Def.	Calculate B and G for disk and drum
RCL	Def.	Calculate B and G for merge-only applications
RCM	Def.	Sort system interface for definition
RCN	Def.	Allocate bin sizes and chose technique for 2314 sort
RCO	Def.	Sort system interface for use with the linkage editor
RCP	Def.	Specify user exits to be link edited
RCQ	Def.	Specify input area for control statements
RCR	Def.	Specify B and G constants using tape
RCS	Def.	Specify B and G codes using tape
RCT	Sort, Int. Merge, & Fin. Merge	Frees storage between phases

(Part 4 of 6)

CSECT Name	Phase in Which Used	Purpose for Which Used
RCU	Def. & Opt.	Contains error messages for definition and optimization phases
RCV	Sort, Int. Merge, & Fin. Merge	Sort system interface for processing records
RCW	Def.	Contains messages for record card interpretation
RCX	Def.	Modifications card error messages
RCY	Def.	Read control card error messages
RCZ	Opt.	Sort system interface used after linkage editor
RC1	Opt.	Expand phase-to-phase information area
RC2	Def.	Extract calculations
RC3	Def.	Contains messages required for scan routine
RC4	Opt.	Control system search after linkage editor
RC6	Sort	Sort phase definition
RC7	Int. merge	Intermediate merge definition
RC8	Fin. merge	Final merge or merge-only definition
RC9	Sort, Int. Merge, & Fin. Merge	Load routine for phases
RDB	Sort	Deblock fixed-length records with in-line move
RDC	Sort	Deblock fixed-length records with link to multiple move
RDD	Sort	Deblock fixed-length records containing user exits
RDE	Sort	Deblock variable-length records containing user exits
RDG	Sort	Deblock variable-length records
RDH	Fin.Merge	Deblock fixed-length records for a sort
RDI	Fin.Merge	Deblock variable-length records for a sort
RDJ	Fin.Merge	Deblock for a merge operation
RDL	Int.Merge	Set up deblock area
RDP	Osc. and C-C Sort	Sort deblock fixed-length record<256 bytes
RDQ	Osc. and C-C Sort	Sort deblock fixed-length record>256 bytes
RDR	Osc. and C-C Sort	Sort deblock fixed-length record with user exits
RDS	Osc. and C-C Sort	Sort deblock variable-length record with user exits
RDT	Osc. and C-C Sort	Sort deblock variable length record without user exits
RDX	Fin. Merge	Deblock for read-forward, fixed-length records in tape sort
RGA	Sort	Indicate an input end-of-file - sort phase
RGB	Int.Merge	Read tape - intermediate merge phase
RGC	Int.Merge	Read disk - intermediate merge phase
RGD	Fin.Merge	Read tape - final merge phase
RGE	Fin.Merge	Read disk - final merge phase
RGF	Fin.Merge	Indicate an input end-of-file - merge-only application
RGL	Int. Merge	Read tape forward
RGM	Fin. Merge	Read tape forward
RGO	Int. Merge	Read drum
RGP	Fin. Merge	Read drum
RMA	Sort	Contains all messages for sort phase
RMB	Int.Merge	Contains all messages for merge phase
RMC	Fin.Merge	Contains all messages for final merge and merge only phases
ROA	Sort	Fixed-length replacement with multiple control fields
ROB	Sort	Fixed-length replacement with single control fields
ROC	Sort	Variable-length replacement with multiple control fields
ROD	Sort	Variable-length replacement with single control fields
ROE	Sort	Using polyphase technique for fixed-length records with multiple control fields
ROF	Sort	Using polyphase technique for fixed-length records with single control fields
ROG	Sort	Using polyphase technique for variable-length records with multiple control fields

(Part 5 of 6)

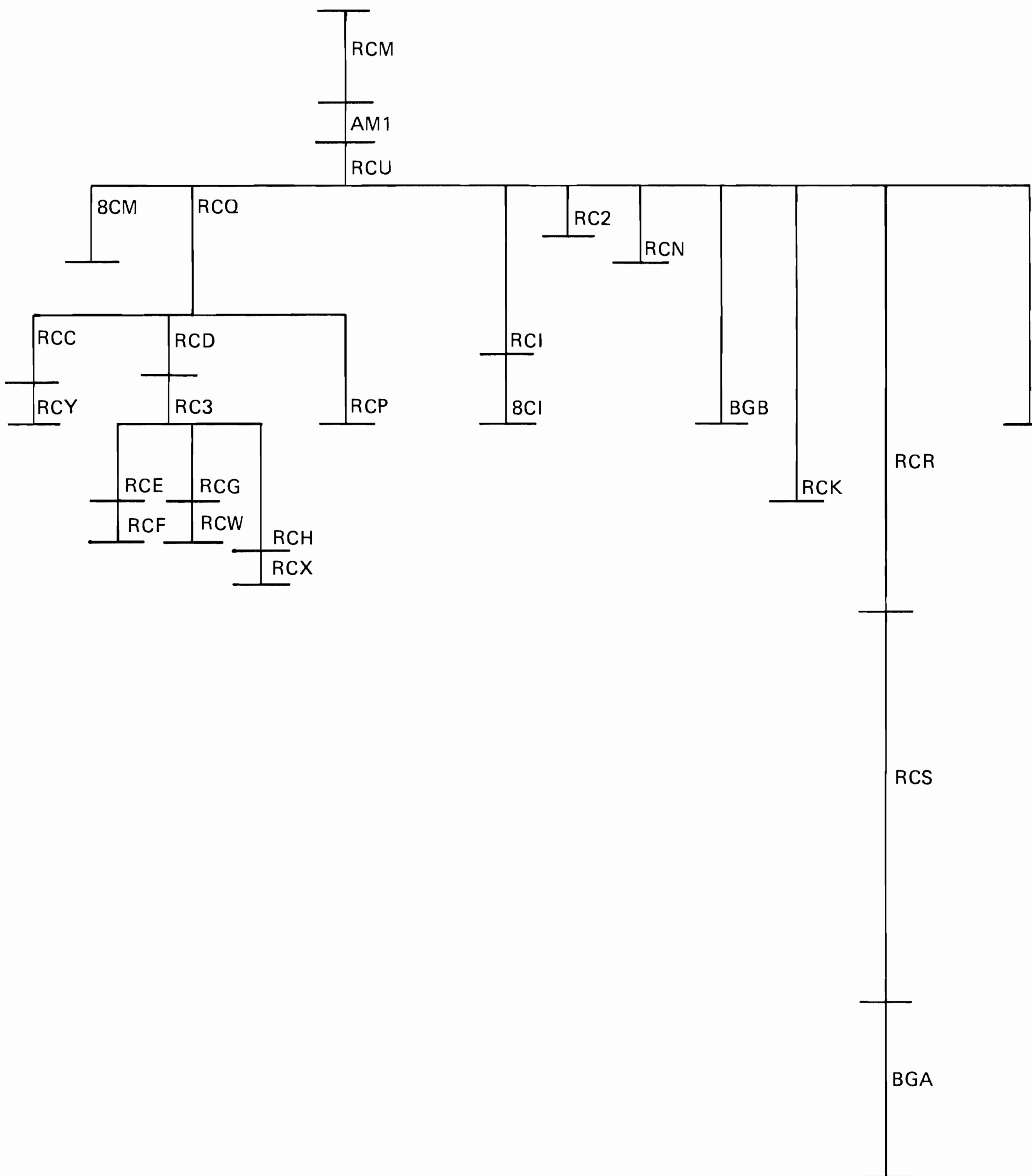
CSECT Name	Phase in Which Used	Purpose for Which Used
ROH	Sort	Using polyphase technique for variable-length records with single control fields
ROI	Sort	Balanced tape sort algorithm
ROJ	Sort	Polyphase sort algorithm
ROK	Sort	Balanced disk sort algorithm
RON	Osc. Sort	Oscillating sort algorithm; initiates checkpoint operations
ROO	Sort	Drum-sort algorithm
ROP	Int. & Fin. Merge	16-way merge network with multiple control fields
ROQ	Int. & Fin. Merge	16-way merge network with single control fields
ROR	Int.Merge	Balanced tape merge algorithm
ROS	Int.Merge	Polyphase tape merge algorithm; initiates checkpoint operations
ROT	Int.Merge	Balanced disk merge algorithm; initiates checkpoint operations
ROU	Int. & Fin. Merge	8-way merge network with multiple control fields
ROV	Int. & Fin. Merge	8-way merge network with single control fields
ROW	Osc. and C-C Sort	Fixed-length records with multiple control fields
ROX	Osc. and C-C Sort	Fixed-length records with single control fields
ROY	Osc. and C-C Sort	Variable-length records with multiple control fields
ROZ	Osc. and C-C Sort	Variable-length records with single control fields
RO3	Int. Merge	Drum-merge algorithm
RPA	Sort	Write tape - sort phase
RPB	Sort	Write disk - sort disk
RPC	Sort	Indicate an end of housekeeping procedures in the sort phase
RPD	Int.Merge	Write tape - intermediate merge phase
RPE	Int.Merge	Write disk - intermediate merge phase
RPF	Int.Merge	Indicate an end of housekeeping procedures in the intermediate merge phase
RPG	Fin.Merge	Indicate an end of housekeeping procedures and open output in the final merge phase
RPM	Osc. Sort	End of phase housekeeping for oscillating sort
RPN	Sort	Write drum
RPO	Int. Merge	Write drum

(Part 6 of 6)

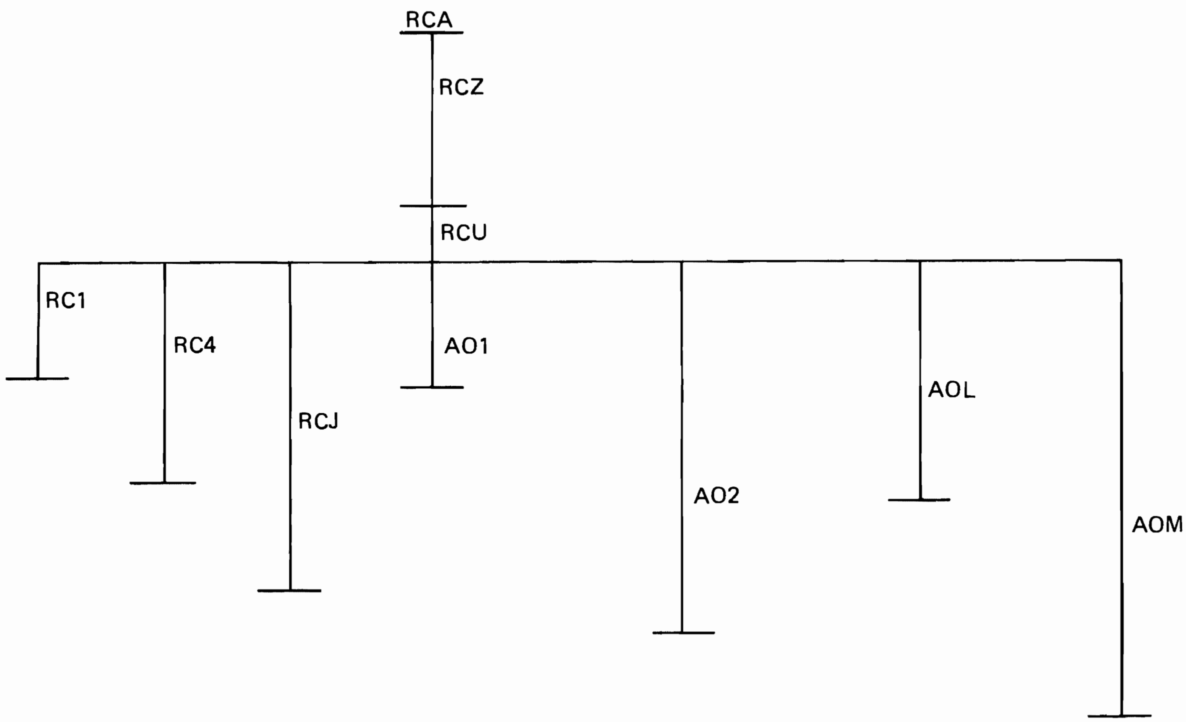
Section 5: Detailed Layouts

Overlay Structure

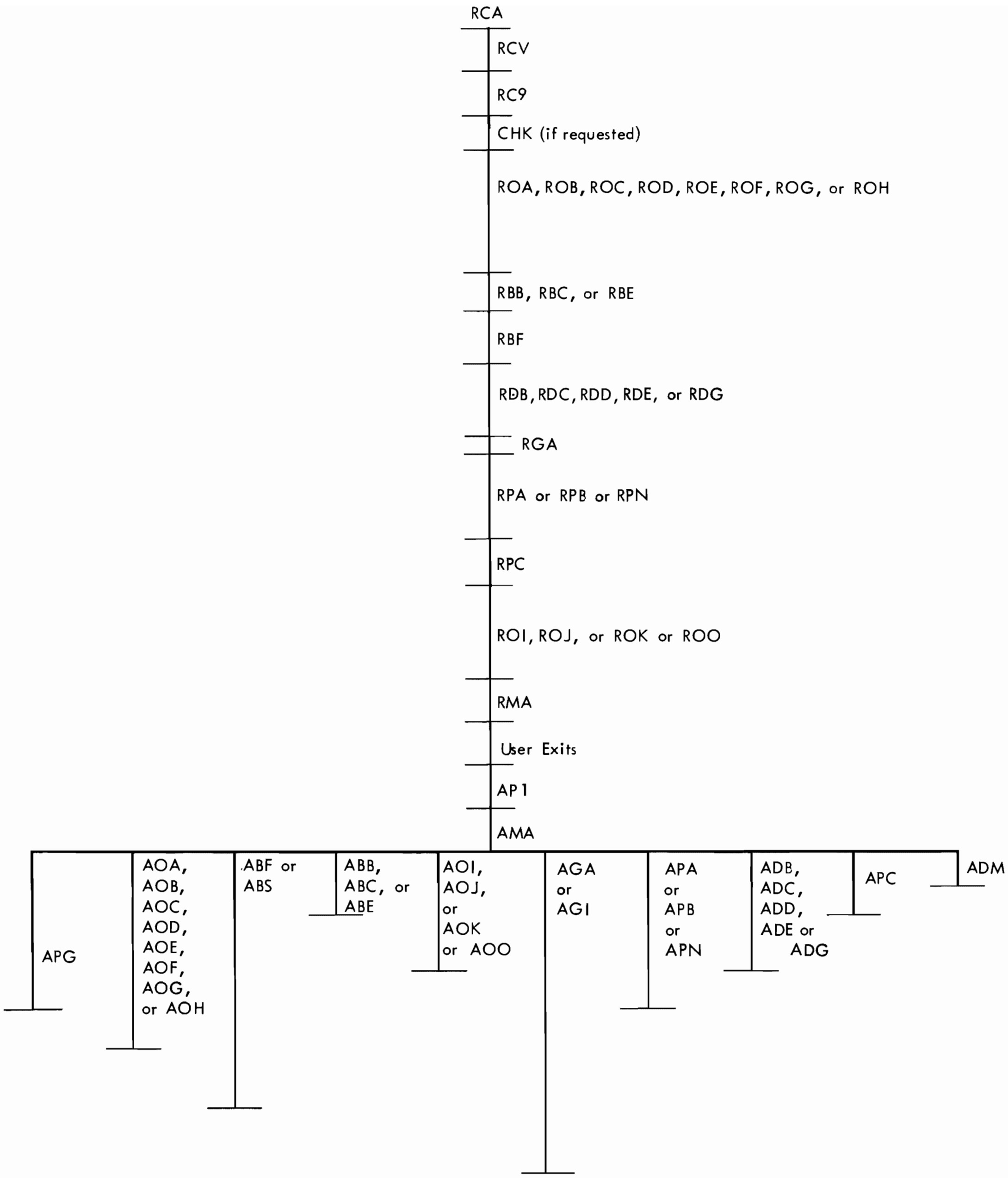
This topic illustrates the overlay structure of the five phases of the sort/merge program. In the multiprogramming environments the same modules are used but may not be placed in contiguous locations.



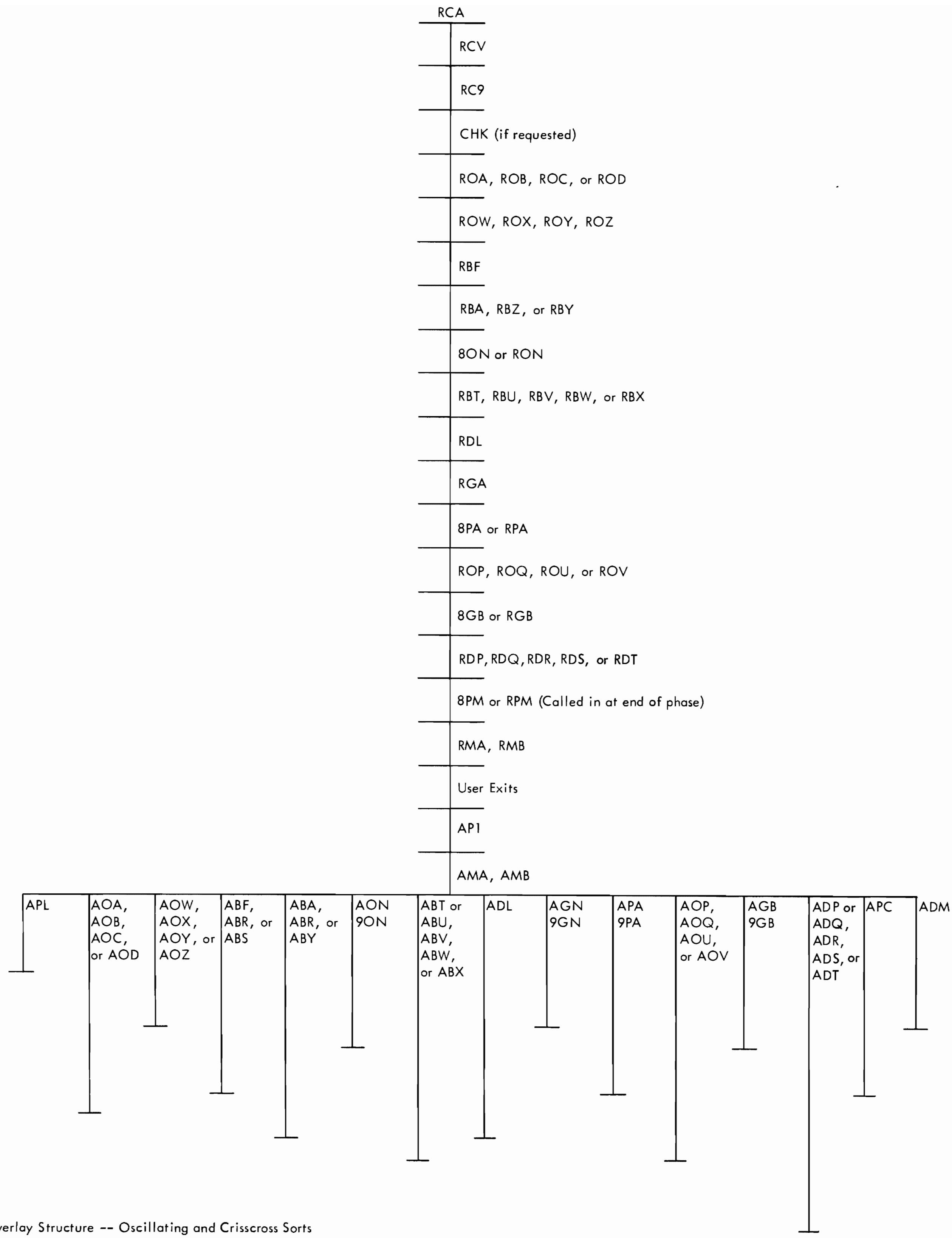
Overlay Structure -- Definition Phase



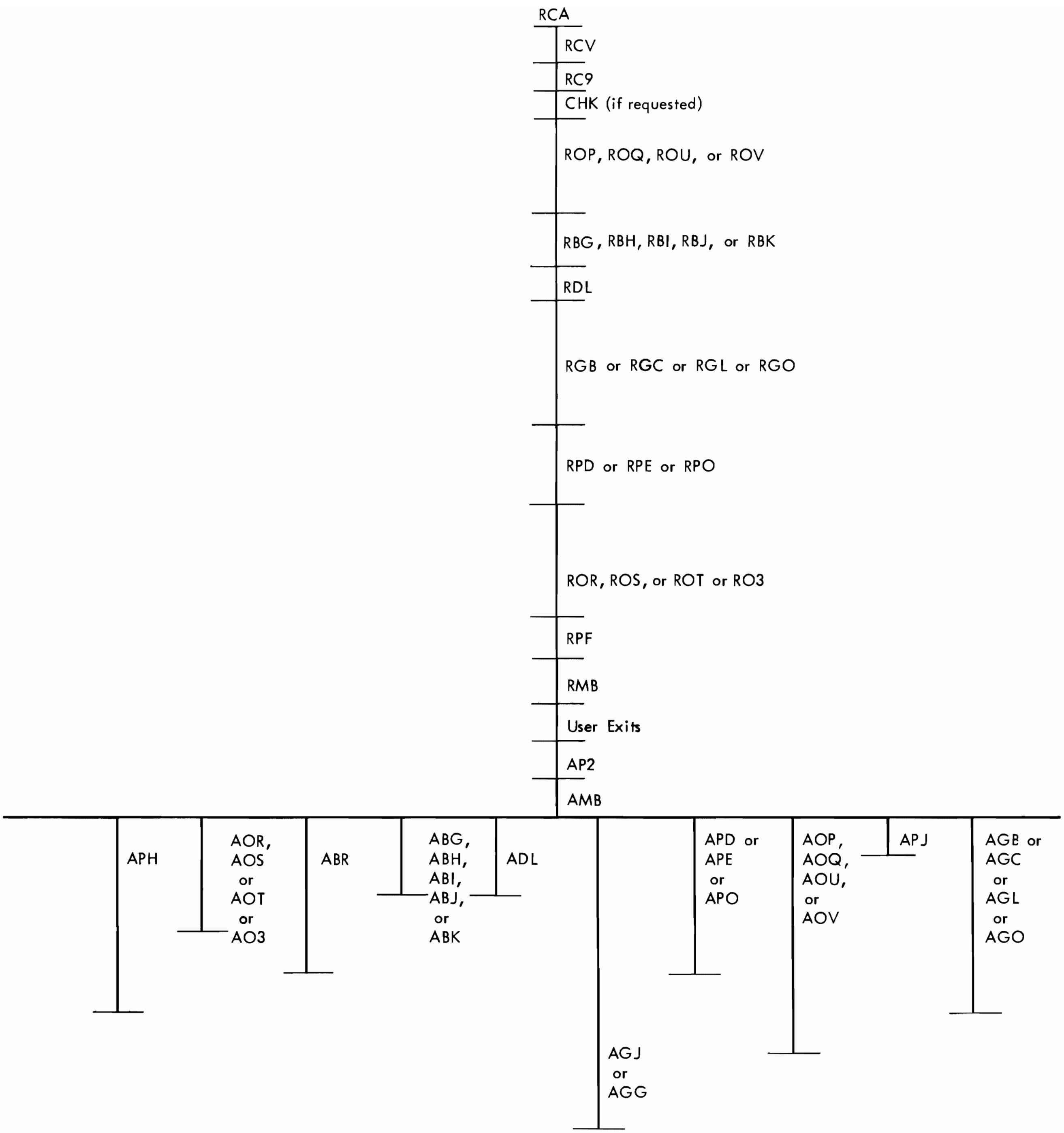
Overlay Structure -- Optimization Phase



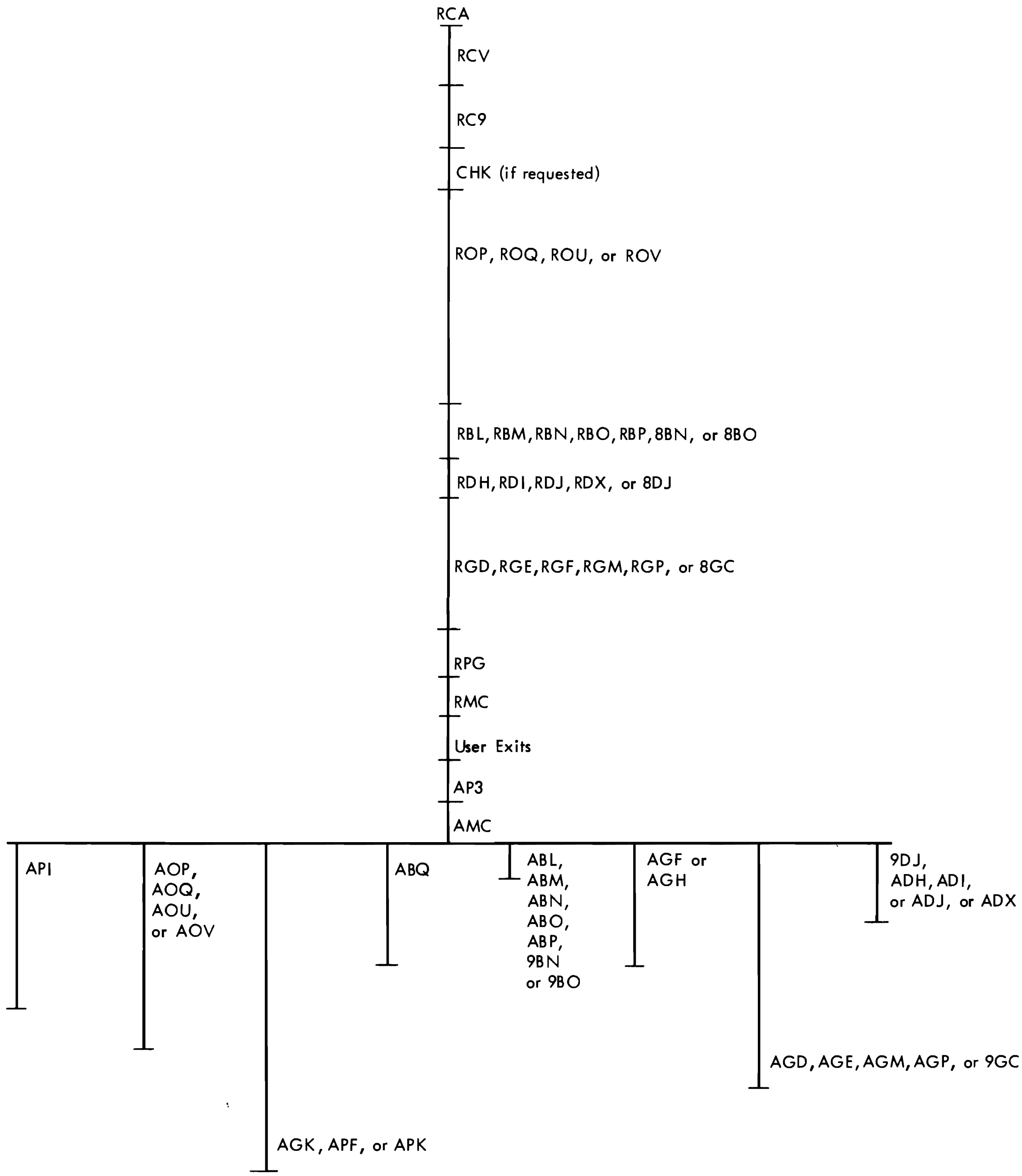
Overlay Structure -- Sort Phase (Not applicable to Oscillating and Crisscross Sorts)



Overlay Structure -- Oscillating and Crisscross Sorts



Overlay Structure -- Intermediate Merge Phase (Not applicable to Oscillating and Crisscross Sorts)



Overlay Structure -- Final Merge Phase

Storage Layouts

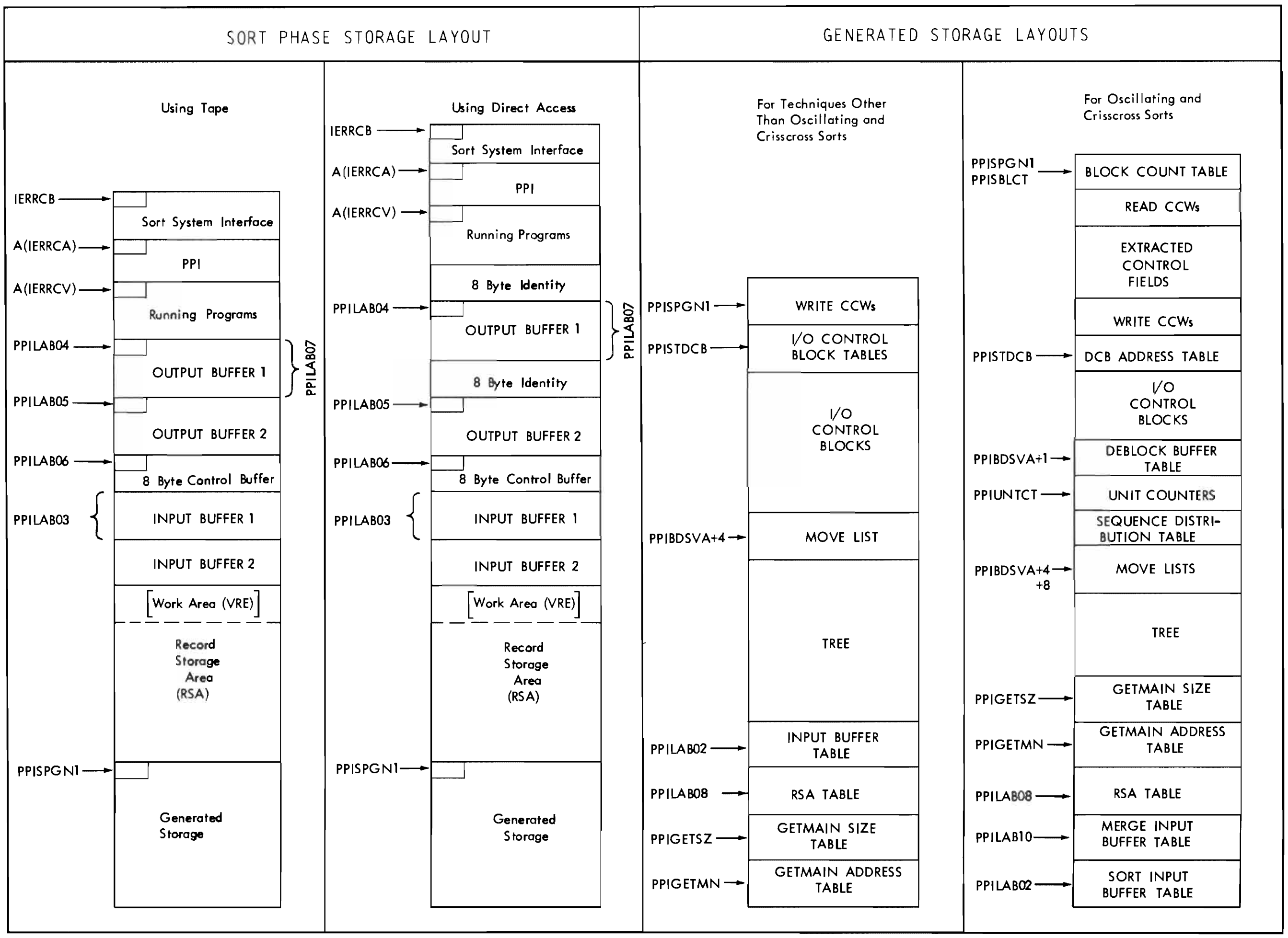
This topic illustrates the main storage layout of the sort, intermediate merge, and final merge phases of the sort/merge program. The labels in these figures represent the pointers located in the PPI area. If user routines are included and a listing of the module map produced by linkage editor is desired, the following DD statement must be included in the job step used to execute the sort:

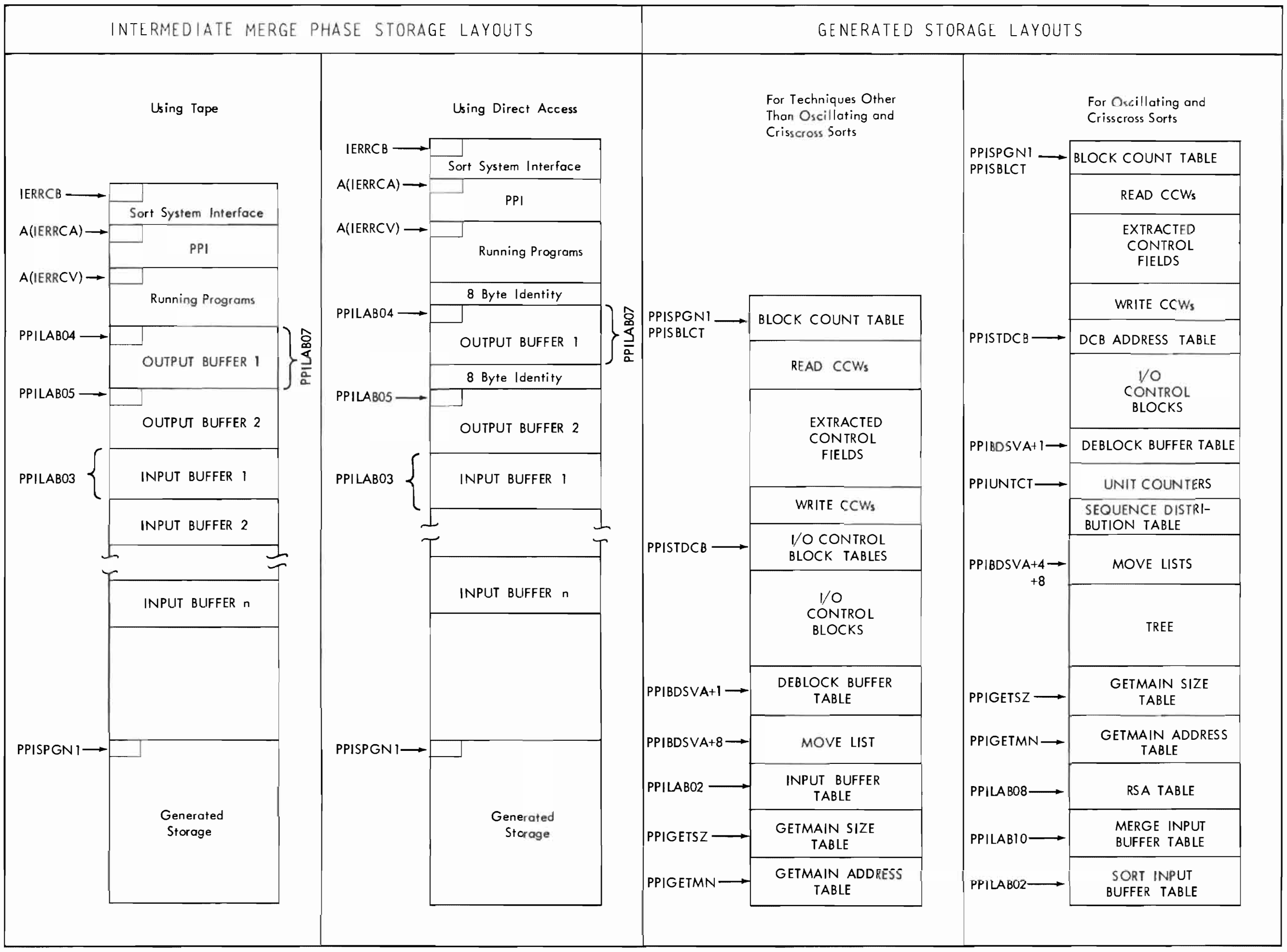
//SORT.SYSPRINT DD SYSOUT=A

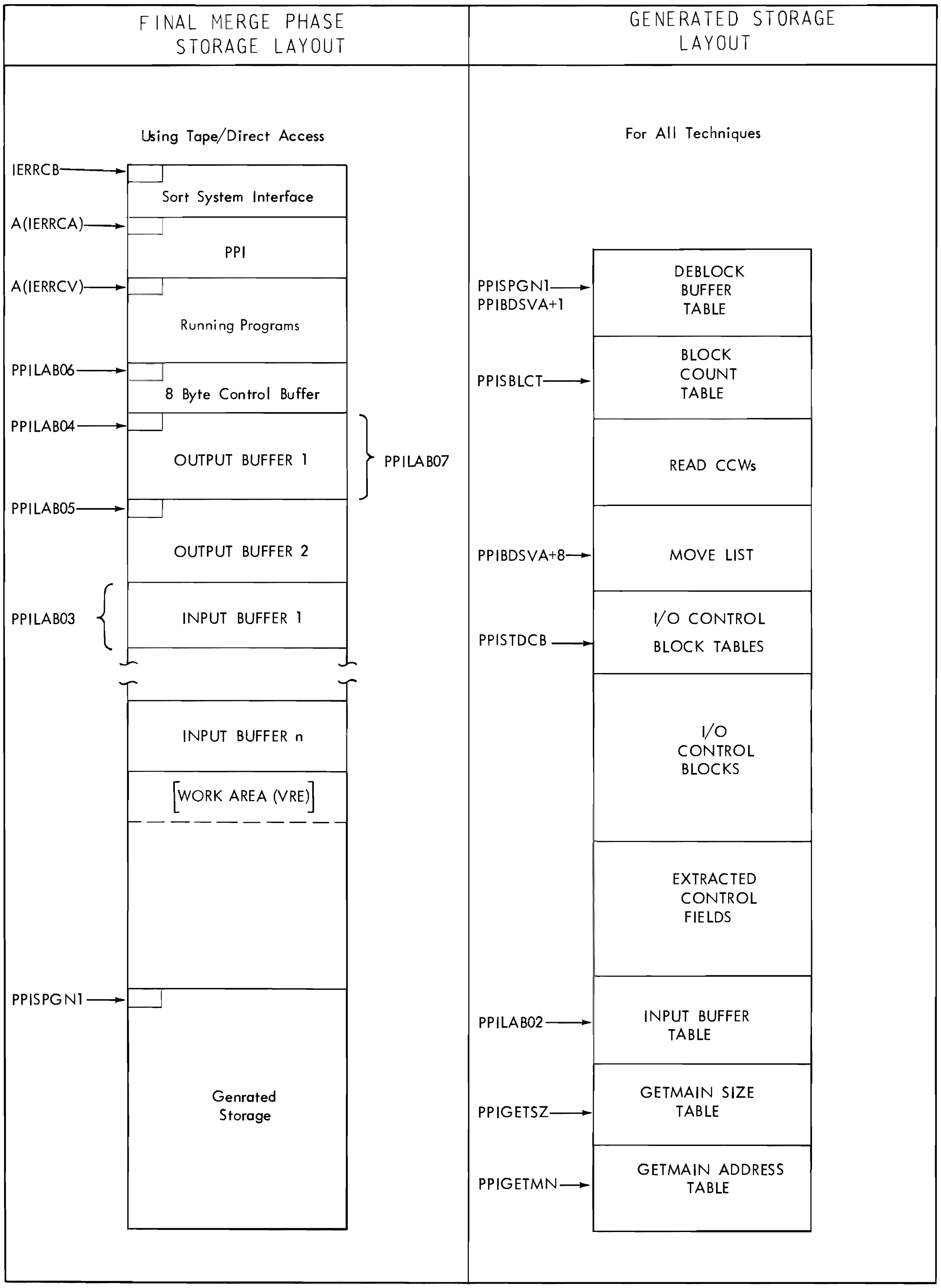
This will override the //SYSPRINT DD DUMMY statement in the SORT cataloged procedure, and a module map from the linkage editor will be written on the data set SYSPRINT.

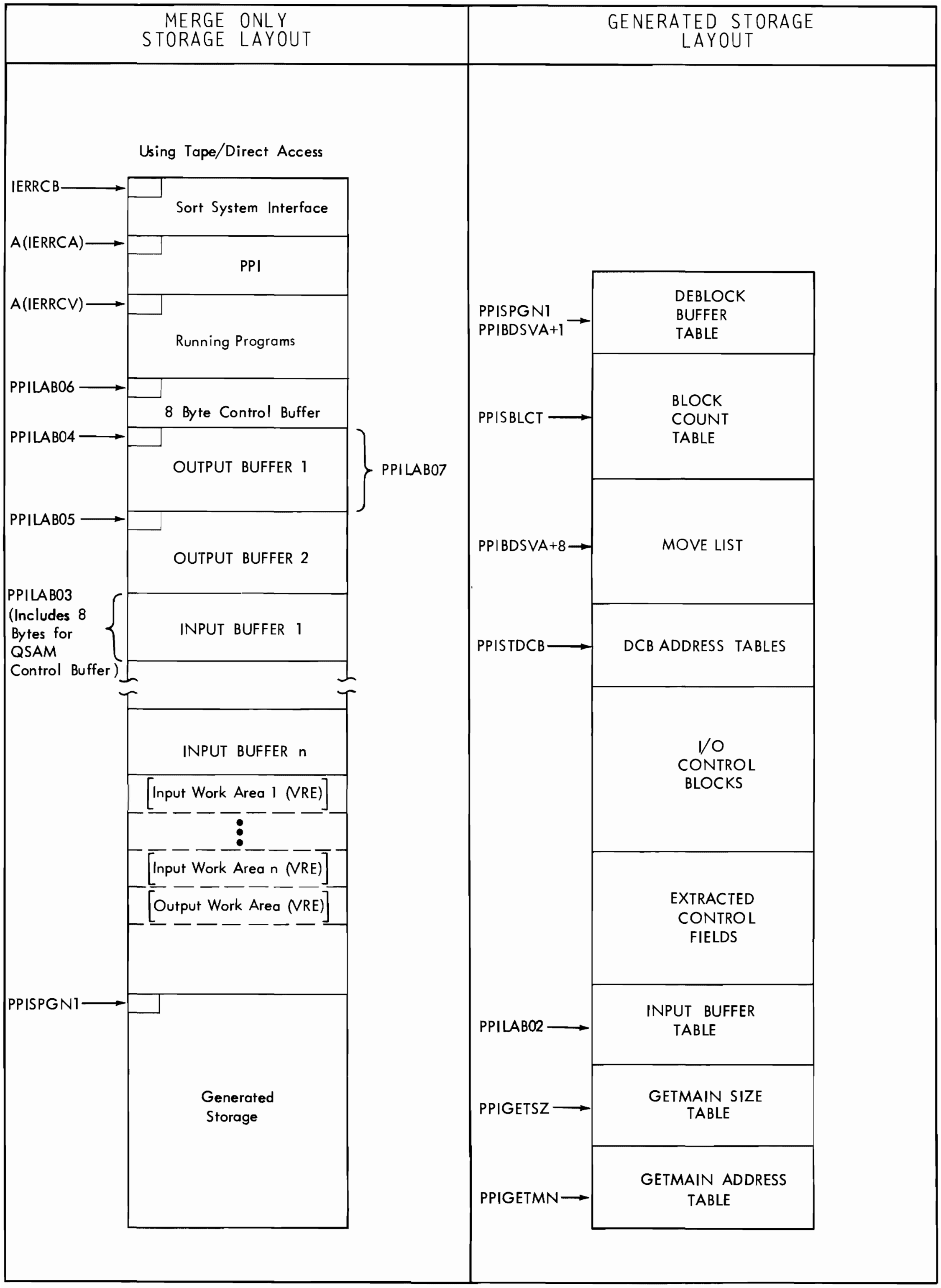
A module map may also be obtained, under any conditions, through use of the DIAG parameter discussed in Appendix C.

Note that the storage layouts illustrated below may not apply when the program is operating in a multiprogramming environment.




INTERMEDIATE MERGE PHASE STORAGE LAYOUTS
GENERATED STORAGE LAYOUTS
Using Tape
IERRCB
Sort System Interface
A(IERRCA)
PPI
A(IERRCV)
Running Programs
PPILAB04
OUTPUT BUFFER 1
PPILAB07
PPILAB05
OUTPUT BUFFER 2
PPILAB03
INPUT BUFFER 1
INPUT BUFFER 2
INPUT BUFFER n
PPISPGN1
Generated Storage
Using Direct Access
IERRCB
Sort System Interface
A(IERRCA)
PPI
A(IERRCV)
Running Programs
8 Byte Identity
PPILAB04
OUTPUT BUFFER 1
PPILAB07
8 Byte Identity
PPILAB05
OUTPUT BUFFER 2
PPILAB03
INPUT BUFFER 1
INPUT BUFFER n
PPISPGN1
Generated Storage
For Techniques Other Than Oscillating and Crisscross Sorts
PPISPGN1
PPISBLCT
BLOCK COUNT TABLE
READ CCWs
EXTRACTED CONTROL FIELDS
WRITE CCWs
PPISTDCB
I/O CONTROL BLOCK TABLES
I/O CONTROL BLOCKS
PPIBDSVA+1
DEBLOCK BUFFER TABLE
PPIBDSVA+8
MOVE LIST
PPILAB02
INPUT BUFFER TABLE
PPIGETSZ
GETMAIN SIZE TABLE
PPIGETMN
GETMAIN ADDRESS TABLE
For Oscillating and Crisscross Sorts
PPISPGN1
PPISBLCT
BLOCK COUNT TABLE
READ CCWs
EXTRACTED CONTROL FIELDS
WRITE CCWs
PPISTDCB
DCB ADDRESS TABLE
I/O CONTROL BLOCKS
PPIBDSVA+1
DEBLOCK BUFFER TABLE
PPIUNTCT
UNIT COUNTERS
SEQUENCE DISTRIBUTION TABLE
PPIBDSVA+4
+8
MOVE LISTS
TREE
PPIGETSZ
GETMAIN SIZE TABLE
PPIGETMN
GETMAIN ADDRESS TABLE
PPILAB08
RSA TABLE
PPILAB10
MERGE INPUT BUFFER TABLE
PPILAB02
SORT INPUT BUFFER TABLE


FINAL MERGE PHASE
STORAGE LAYOUT
GENERATED STORAGE
LAYOUT
Using Tape/Direct Access
For All Techniques
IERRCB
Sort System Interface
A(IERRCA)
PPI
A(IERRCV)
Running Programs
PPILAB06
8 Byte Control Buffer
PPILAB04
OUTPUT BUFFER 1
PPILAB07
PPILAB05
OUTPUT BUFFER 2
PPILAB03
INPUT BUFFER 1
INPUT BUFFER n
[WORK AREA (VRE)]
PPISPGN1
Genrated
Storage
PPISPGN1
PPIBDSVA+1
DEBLOCK
BUFFER
TABLE
PPISBLCT
BLOCK
COUNT
TABLE
READ CCWs
PPIBDSVA+8
MOVE LIST
PPISTDCB
I/O CONTROL
BLOCK TABLES
I/O
CONTROL
BLOCKS
EXTRACTED
CONTROL
FIELDS
PPILAB02
INPUT BUFFER
TABLE
PPIGETSZ
GETMAIN SIZE
TABLE
PPIGETMN
GETMAIN ADDRESS
TABLE


MERGE ONLY STORAGE LAYOUT
GENERATED STORAGE LAYOUT
Using Tape/Direct Access
IERRCB
Sort System Interface
A(IERRCA)
PPI
A(IERRCV)
Running Programs
PPILAB06
8 Byte Control Buffer
PPILAB04
OUTPUT BUFFER 1
PPILAB07
PPILAB05
OUTPUT BUFFER 2
PPILAB03 (Includes 8 Bytes for QSAM Control Buffer)
INPUT BUFFER 1
INPUT BUFFER n
Input Work Area 1 (VRE)
Input Work Area n (VRE)
Output Work Area (VRE)
PPISPGN1
Generated Storage
PPISPGN1
PPIBDSVA+1
DEBLOCK BUFFER TABLE
PPISBLCT
BLOCK COUNT TABLE
PPIBDSVA+8
MOVE LIST
PPISTDCB
DCB ADDRESS TABLES
I/O CONTROL BLOCKS
EXTRACTED CONTROL FIELDS
PPILAB02
INPUT BUFFER TABLE
PPIGETSZ
GETMAIN SIZE TABLE
PPIGETMN
GETMAIN ADDRESS TABLE

Phase-to-Phase Information (PPI) Area

This topic contains a storage map of the phase-to-phase information (PPI) area. PPI is a communication area for all modules of the sort/merge program. It is created during the definition phase and resides in main storage throughout the execution of a sort. PPI is not executable and has the name IERRCA. An explanation of each field within PPI is contained in the program listing of module IERRCA.

The sort/merge program uses general purpose register 13 as the base register for PPI. Reference to any field within PPI can be made by adding the appropriate displacement to the contents of register 13. The displacement for each field within PPI appears in hexadecimal and decimal form to the left of the storage map. Byte counts within parentheses are in decimal form.

Displ. Hex.	Displ. Dec.	← 8 BYTES →				
0	0	PPISVARE				
48	72	PPIWKARE (256 bytes) PPIPDWA (64 bytes) PPIP1GC (4 bytes)	PPIP1ASZ			
50	80	PPISKPRD	PPIATP1E			
58	88	PPIIPBLK	PPIBUF1			
60	96	PPIEXTSZ	PPIFFF			
68	104	PPIPBUFF	Reserved for Future Use			
70	112	PPINUMCF	PPIPCF01	PPIMCF01	PPIFCF01	
78	120	PPIPCF02	PPIMCF02	PPIFCF02	PPIPCF03	
80	128	PPIPCF03 (cont)	PPIMCF03	PPIFCF03	PPIPCF04	PPIMCF04
88	136	PPIMCF04 (cont) PPIPSVA (64 bytes)	PPIFCF04	PPIPCF05	PPIMCF05	PPIFCF05
90	144	PPIPCF06	PPIMCF06	PPIFCF06	PPIPCF07	
98	152	PPIPCF07 (cont) PPIDSKED (136 bts.)	PPIMCF07	PPIFCF07	PPIPCF08	PPIMCF08
A0	160	PPIMCF08 (cont)	PPIFCF08	PPIPCF09	PPIMCF09	PPIFCF09
A8	168	PPIPCF10	PPIMCF10	PPIFCF10	PPIPCF11	
B0	176	PPIPCF11 (cont)	PPIMCF11	PPIFCF11	PPIPCF12	PPIMCF12
B8	184	PPIMCF12 (cont)	PPIFCF12			
C0	192	(END OF "PPIPSVA" ALLOCATION)				
C8	200					

Hex	Decimal	Contents
C8	200	PPILAB01
		PPITPPT
F8	248	PPITPTBL (136 bytes)
118	280	PPIDIRAD (END OF "PPIDSKED" ALLOCATION)
120	288	PPISTAR (136 bytes), PPIODOM (64 bytes) (END OF "PPIWKARE" ALLOCATION) (END OF "PPIODOM" ALLOCATION) (END OF "PPITPTBL" ALLOCATION)
180	384	(END OF "PPISTAR" ALLOCATION)
1A8	424	PPIENDAR (136 bytes)
		(END OF "PPIENDAR" ALLOCATION)
230	560	

Hex	Dec	+0 – +1	+2 – +3	+4 – +5	+6 – +7
230	560	PPISW1 +1	+2 +3	+4 +5	+6 +7
238	568	PPIMODEX		PPILINK	
240	576	PPICOUNT		PPIDELCT	
248	584	PPIINSCT		PPIRCDCT	
250	592	PPISEQCT			
				PPIFILSZ	
260	608	PPIBINSZ		PPINMAX	
268	616	PPIRMAX		PPISRTG	
270	624	PPISRTBL	PPIOPBLK	PPIBUF23	Reserved
278	632	PPIOPFMP			
280	640	PPIDEPHO			
288	648	PPIRCDL1	PPIRCDL2	PPIRCDL3	PPIRCDL4
290	656	PPIRCDL5	PPIMRGMX	PPIMRGAL	PPIMRGOP
298	664	PPIDDOL1		PPIAXERT	
2A0	672	PPIUSER		PPILEXFD	PPILEXFF
2A8	680	PPINDSKA	PPIBPTRK	PPILAB03	
2B0	688	PPILAB07		PPIDOUO (4 bytes), PPILAB09	
2B8	696	PPIP2GC		PPIP3GC	
2C0	704	PPIP3ASZ		PPIATP3E	
2C8	712	PPITAVLC		PPITREND	
2D0	720	PPISPGN1		PPILAB02	
2D8	728	PPILAB04		PPILAB05	
2E0	736	PPILAB06		PPIDOOBA (4 bytes), PPILAB08	
2E8	744	PPIBDSVA			
2F8	760	PPISTDCB		PPISBLCT	
300	768	PPISTIOB		PPIUNTCT	
308	776	PPILAB10		PPIGETMN	
310	784	PPIGETSZ		PPISORCE	
318	792	PPISORCE (cont)		PPISLIB	
320	800	PPIRCV		PPIADSSC	
328	808	PPIALG			
330	816	PPIDEB			
338	824	PPINET			
340	832	PPIBLK			
348	840	PPIWRT			
350	848	PPIVMV			
358	856	PPIRD			
360	864	PPIDEB2			
368	872	PPINETM			
370	880	PPIBLK2			
378	888	PPIINT			
380	896	PPICONV			
388	904	PPIEOF			
390	912	PPIRMA			
398	920	PPIRMB, PPIRMC (8 bytes)			
3A0	928	PPIAMA			
3A8	936				

3A8	936	PPIAMB, PPIAMC (8 bytes)	
3B0	944	PPIOPEN	
3B8	952	PPIX11	
3C0	960	PPIX21, PPIX31 (8 bytes)	
3C8	968	PPIX15	
3D0	976	PPIX25, PPIX35 (8 bytes)	
3D8	984	PPIX17	
3E0	992	PPIX27, PPIX37 (8 bytes)	
3E8	1000	PPIX18	
3F0	1008	PPIX28, PPIX38 (8 bytes)	
3F8	1016	PPIX19	
400	1024	PPIX29, PPIX39 (8 bytes)	
408	1032	PPIX61	
410	1040	PPIX16	
418	1048	PPIADDCF	PPIDDSRT ("SORT")
420	1056	VER. # Not used	PPICHKAD

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
0	0	PPISVARE	72	Register save area
48	72	PPIWKARE	256	Starting address of sort work area
		PPIPDWA	64	Merge network prime area
		PPIP1GC	4	Size of sort phase generated core
4C	76	PPIP1ASZ	4	Phase 1 assignment size
50	80	PPISKPRD	4	Skip record count
54	84	PPIATP1E	4	Address of ATTACHors phase 1 exit
58	88	PPIIPBLK	4	Input blocking
5C	92	PPIBUF1	4	Number of buffers -- phase 1
60	96	PPIEXTSZ	4	Size of extract routine
64	100	PPIFFF	4	Displacement of F field
68	104	PPIPBUFF	4	Displacement of packing buffer
6C	108	reserved	4	
70	112	PPINUMCF	2	Number of control fields
72	114	PPIPCF01	3	Control field 1 position
75	117	PPIMCF02	2	Control field 1 length
77	119	PPIFCF01	1	Control field 1 format and sequence
78	120	PPIPCF02	3	Control field 2 position
7B	123	PPIMCF02	2	Control field 2 length
7D	125	PPIFCF02	1	Control field 2 format and sequence
7E	126	PPIPCF03	3	Control field 3 position
81	129	PPIMCF03	2	Control field 3 length
83	131	PPIFCF03	1	Control field 3 format and sequence
84	132	PPIPCF04	3	Control field 4 position
87	135	PPIPMCF04	2	Control field 4 length
88	136	PPIPSVA	64	Merge network prime save area
89	137	PPIFCF04	1	Control field 4 format and sequence
8A	138	PPIPCF05	3	Control field 5 position
8D	141	PPIMCF05	2	Control field 5 length
8F	143	PPIFCF05	1	Control field 5 format and sequence
90	144	PPIPCF06	3	Control field 6 position
93	147	PPIMCF06	2	Control field 6 length
95	149	PPIFCF06	1	Control field 6 format and sequence
96	150	PPIFCF07	3	Control field 7 position

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
98	152	PPIDSKED	136	Used for up to 17 disk addresses. During general assignment, the address format is ABC A = 1 byte channel address B = 1 byte unit address C = 2 bytes, number of tracks During running program, the address format is M BB CC HH R
99	153	PPIMCF07	2	Control field 7 length
9B	155	PPIFCF07	1	Control field 7 format and sequence
9C	156	PPIPCF08	3	Control field 8 position
9F	159	PPIMCF08	2	Control field 8 length
A1	161	PPIFCF08	1	Control field 8 format and sequence
A2	162	PPIPCF09	3	Control field 9 position
A5	165	PPIMCF09	2	Control field 9 length
A7	167	PPIFCF09	1	Control field 9 format and sequence
A8	168	PPIPCF10	3	Control field 10 position
AB	171	PPIMCF10	2	Control field 10 length
AD	173	PPIFCF10	1	Control field 10 format and sequence
AE	174	PPIPCF11	3	Control field 11 position
B1	177	PPIMCF11	2	Control field 11 length
B3	179	PPIFCF11	1	Control field 11 format and sequence
B4	180	PPIPCF12	3	Control field 12 position
B7	183	PPIMCF12	2	Control field 12 length
B9	185	PPIFCF12	1	Control field 12 format and sequence
C8	200	PPILAB01	64	Utility storage position, used for read/write directory
F4	244	PPITPPT	4	Tape table pointer -- high order byte is channel of SORTIN
F8	248	PPITPTBL	136	Tape table -- two byte entry for each unit of the form

$x_1x_2x_3x_4$0000	DCB Increment

x_1 - I/O bit
1 = input
0 = output

x_2 - Open bit
1 = open routine should open unit

x_3 - Full reel bit (oscl only)
1 = unit contains full reel
0 = unit does not contain full reel

x_4 - Collating sequence bit (oscl only) set at EOF or RMAX time only
1 = descending
0 = ascending

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
118	280	PPIDIRAO	8	Disk directory address
120	288	PPIODOM	64	Odometer table for oscl sort, one word for each of 16 levels Byte 0 = number of sequences at this level Bytes 1-3 = Address of next tape table entry to be used as output for this level
120	288	PPISTAR	136	Direct access starting addresses, one entry for each of 3-17 extents
1A8	424	PPIENDAR	136	Direct access ending addresses reset from PPIDSKED for 2301-2311 Phase 3 starting disk addresses for 2314 read priming
230	560	PPISW1	8	Switch -- 64 bits Bit 0 fixed -- used by read/write rtns Bit 1 variable -- used by read/write rtns If on, the following bits mean: Bit 2 single control field Bit 3 multiple control field Bit 4 balanced Bit 5 polyphase Bit 6 oscillating Bit 7 1 to 8 Bit 8 1 to 16 Bit 9 tape Bit 10 disk Bit 11 no data chaining Bit 12 data chaining input Bit 13 data chaining output Bit 14 MODS Bit 15 no MODS Bit 16 records .LT. 256 Bit 17 records .GT. 256 Bit 18 skip option Bit 19 phase 1 Bit 20 phase 2 Bit 21 phase 3 Bit 22 merge only Bit 23 checkpoint Bit 24 equals Bit 25 extract Bit 26 user's output sequence = 1 - descending = 0 - ascending Bit 27 phase 1 collating order or merge input order = 1 descending = 0 ascending Bit 28 disk merge table collating order = 1 descending = 0 ascending Bit 29 attached, linked, or executed Bit 30 filesize estimated = 1 - estimated = 0 - not specified Bit 31 merge only - assignment or running program EOF = 1 - running program EOF = 0 - assignment EOF oscillating QSAM has detected EOF = 1 QSAM EOF = 0 QSAM has not detected EOF

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
				Bit 32 E type control fields present Bit 33 SORT card present Bit 34 MERGE card present Bit 35 RECORD card present Bit 36 MODS card present Bit 37 execute entire system search Bit 38 doubleword alignment for buffers Bit 39 single word alignment for buffers Bit 40 read error flag Bit 41 write error flag Bit 42 even/odd switch for baln Bits 43-44 00 = 1 mpx 01 = 1 mpx and 1 sel, or 1 sel 10 = 1 mpx and N sel, or N sel Bit 45 input unit used as work unit Bit 46 switch or TAU Bit 47 N channel environment Bit 48 deblock backward Bit 49 read forward Bit 50 close with rewind Bit 51 block forward Bit 52 read forward later Bit 53 drum Bit 54 2314 using crcx Bit 55 diagnostic = 1 - diagnostics = 0 - no diagnostics Bit 56 user EOF (oscl only) Bit 57 RMAX reached (oscl only) Bit 58 user insert in process (oscl only) Bit 59 track tape = 0-9 track = 1-7 track Bit 60 merge pass to follow -- 2314 Bit 61 2311s using 2314 technique (never on) Bit 62 value count of 256 for FIELDS parameter or = 1 - baln on 2314 = 0 - not baln on 2314 Bit 63 accept/skip option activated
238	568	PPIMODEX	4	Modification exits activated

Bit	Meaning	Bit	Meaning
0	E11	11	E37
1	E15	12	E38
2	E16	13	E61
3	E17	14	E19
4	E18	15	E29
5	E21	16	E39
6	E25	17-21	not used
7	E27	22	VRE on input
8	E28	23	VRE on output
9	E31	24-31	not used
10	E35		

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
23C	572	PPILINK	4	MOD exit link edit information Bits 0-16 represent exits in the order specified in PPIMODEX field 1 = exit rtn was link edited via sort 0 = exit rtn not link edited via sort

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
				Bit — Meaning if = 1 17 — E11 link edited separately 18 — E21 link edited separately 19 — E31 link edited separately 20 — link editing was done 21 — not used 22 — link edit error 23-31 — not used
240	576	PPICOUNT	4	Record counter Phase 1 - oscl count of records deblocked from input data set Phase 2 - count of records written out on work units Phase 3 - count of records placed on SORTOUT
244	580	PPIDELCT	4	Deleted records count
248	584	PPIINSCT	4	Inserted records count
24C	588	PPIRCDCT	4	Record counter -- total records, including inserts, entering a phase
250	592	PPISEQCT	12	Sequence counters
25C	604	PPIFILSZ	4	File size from SIZE parameter on SORT or MERGE control card
260	608	PPIBINSZ	4	Bin size
264	612	PPINMAX	4	Nmax
268	616	PPIRMAX	4	For fixed-length records, number of records at sort blocking that can be contained in a full reel For variable-length records, number of bytes at sort blocking that can be contained in a full reel
26C	620	PPISRTG	4	G -- number records in RSA
270	624	PPISRTBL	2	B -- sort blocking
272	626	PPIOPBLK	2	Output blocking
274	628	PPIBUF23	2	Number of buffers -- phases 2 and 3
276	630	reserved	2	
278	632	PPIOPFMP	8	Output unit for phase 3
280	640	PPIDEPHO	8	Output unit address
288	648	PPIRCDL1	2	Fixed -- input record length Variable -- maximum input record length
28A	650	PPIRCDL2	2	Fixed -- sort record length Variable -- maximum sort record length
28C	652	PPIRCDL3	2	Fixed -- output record length Variable -- maximum output record length
28E	654	PPIRCDL4	2	Fixed -- not used Variable -- minimum sort record length
290	656	PPIRCDL5	2	Fixed -- not used Variable -- modal record length

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
292	658	PPIMRGMX	2	Maximum merge order
294	660	PPIMRGAL	2	Alternate merge order Poly = 1 Oscl = 1 Baln = alternate merge order Disk = maximum merge order saved
296	662	PPIMRGOP	2	Optimum merge order Poly none Baln none Oscl none Disk optimum merge order
298	664	PPIDD0L1	4	Merge network's major control field
29C	668	PPIAXERT	4	Address of equals or extract module
2A0	672	PPIUSER	4	User communication area
2A4	676	PPILEXFD	2	Length of extracted fields
2A6	678	PPILEXFF	2	Length of extracted fields full
2A8	680	PPINDSKA	2	Number of disk areas
2AA	682	PPIBPTRK	2	Blocks/track for direct access
2AC	684	PPILAB03	4	Input buffer size; first byte -- number of input buffers
2B0	688	PPILAB07	4	Output buffer size; first byte -- number of output buffers
2B4	692	PPIDOUO	4	User option for sequence check
2B4	692	PPILAB09	4	Byte 0 - number of phase 2 output buffers Byte 1 - number of phase 3 output buffers Bytes 2-3 - Phase 3 output buffer size
2B8	696	PPIP2GC	4	Size of merge phase generated core
2BC	700	PPIP3GC	4	Size of final merge phase generated core
2C0	704	PPIP3ASZ	4	Message index
2C4	708	PPIATP3E	4	Address of ATTACHor's phase 3 exit
2C8	712	PPITAVLC	4	Sort phase available core
2CC	716	PPITREND	4	Ending address of tree
2D0	720	PPISPGN1	4	Address of next available byte in generated core
2D4	724	PPILAB02	4	Address of input buffer table Oscl -- address of sort phase input buffer table
2D8	728	PPILAB04	4	Address of output buffer 1
2DC	732	PPILAB05	4	Address of output buffer 2
2E0	736	PPILAB06	4	Address of control buffer Phase 1 -- input buffer pool Phase 3 merge-only -- output buffer pool

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
2E4	740	PPIDOOBA	4	Byte 0 - Number of entries in RSA table Bytes 1-3 - Address of RSA table
2E8	744	PPIBDSVA	16	Block/deblock save area Byte 0 - Total number of work units Bytes 1-3 - Address of input buffer table for phases 2 and 3 Bytes 4-7 - Fixed -- address of move list phase 1 Variable -- address of next available bin Bytes 8-11 - Fixed -- Address of move list phases 2 and 3 Variable -- Number of available bins Bytes 12-15 - Variable -- entry to move routine phase 1
2F8	760	PPISTDCB	4	Starting address of DCB table
2FC	764	PPISBLCT	4	Address of block count table
300	768	PPISTIOB	4	Starting address of IOB table
304	772	PPIUNTCT	4	Oscl only -- address of unit count table
308	776	PPILAB10	4	Oscl only -- address of input buffer table for merge phase
30C	780	PPIGETMN	4	Address of GETMAIN table of addresses
310	784	PPIGETSZ	4	Address of GETMAIN table of sizes
314	788	PPISORCE	8	ddname of user mod library DCB addresses of SYSLMOD user library
31C	796	PPISLIB	4	DCB addresses of sort library
320	800	PPIRCV	4	Sort system control for running program
324	804	PPIADSSC	4	

Module interface list, Format of each entry is:

4 bytes	4 bytes
3 character symbolic names	absolute address

Hex.	Dec.	Field Name	Bytes	Field Description
328	808	PPIALG	8	Algorithm phases 1 and 2
330	816	PPIDEB	8	Deblock phases 1 and 3
338	824	PPINET	8	Network phases 1 and 3
340	832	PPIBLK	8	Block phases 1 and 3
348	840	PPIWRT	8	Write phases 1 and 2
350	848	PPIVMV	8	Variable move -- sort phase
358	856	PPIRD	8	Read phases 2 and 3
360	864	PPIDEB2	8	Deblock phase 2 prime routine
368	872	PPINETM	8	Merge network phase 2
370	880	PPIBLK2	8	Block/deblcok phase 2
378	888	PPIINT	8	Initialize sort and tree, oscl

Displacement Hex.	Dec.	Field Name	Bytes	Field Description
380	896	PPICONV	8	Convert hex to characters for message
388	904	PPIEOF	8	EODAD for QSAM phase 1 and merge only
390	912	PPIRMA	8	Messages for phase 1 running program
398	920	PPIRMC	8	Messages for phase 3 running program
398	920	PPIRMB	8	Messages for phase 2 running program
3A0	928	PPIAMA	8	Messages for phase 1 assignment prog
3A8	936	PPIAMC	8	Messages for phase 3 assignment prog
3A8	936	PPIAMB	8	Messages for phase 2 running prog
3B0	944	PPIOPEN	8	Open list for phases 1, 2, and 3
3B8	952	PPIX11	8	Exits for user initialization functions
3C0	960	PPIX31	8	
3C0	960	PPIX21	8	
3C8	968	PPIX15	8	Exits for logical record modification
3D0	976	PPIX35	8	
3D0	976	PPIX25	8	
3D8	984	PPIX17	8	Exits for closing data sets at end of phase
3E0	992	PPIX37	8	
3E0	992	PPIX27	8	
3E8	1000	PPIX18	8	Exits for read errors
3F0	1008	PPIX38	8	
3F0	1008	PPIX28	8	
3F8	1016	PPIX19	8	Exits for write errors
400	1024	PPIX39	8	
400	1024	PPIX29	8	
408	1032	PPIX61	8	Exit for extract
410	1040	PPIX16	8	Exit for Nmax
418	1048	PPIADDCF	4	Address of control field info for more than 12 control fields
41C	1052	PPIDDSRT	4	Four letter identification from EXEC statement PARM field - - used when sort is linked to or attached
420	1056		1	PPI version number
421	1057			Not used
424	1060	PPICHKAD	4	Checkpoint module address
428	1064	PPIDCBIN	2	Size of SORTIN DCB
42A	1068	PPIDCBOU	2	Size of SORTOUT DCB

Appendix A: User Program-Modification Exits

This appendix lists the sort/merge modules that have provisions for exits to a user's modification routine. For a description of the modification exits, see the publication OS Sort/Merge.

User program-modification exits in the sort phase are as follows:

Exit	Module(s)
E11	IERAPG, IERAPL
E15	IERRDD, IERRDE, IERRDR, IERRDS
E16	IERRDD, IERRDE, IERRDR, IERRDS
E17	IERRPC, IERRPM, IER8PM
E18	IERAGA, IERAGI, IERAGN, IER9GN
E19	IERAGA, IERAGI, IERAGN, IER9GN
E61	IERROB, IERROD, IERROF, IERROH

User program-modification exits in the intermediate merge phase are as follows:

Exit	Module(s)
E21	IERAPH, IERAPL
E25	IERRBJ, IERRBK, IERRBW, IERRBX
E27	IERRPF, IERRPM, IER8PM
E28	IERAGG, IERAGJ, IERAGN, IER9GN
E29	IERAGG, IERAGJ, IERAGN, IER9GN
E61	IERROQ, IERROV

User program-modification exits in the final merge phase are as follows:

Exit	Module(s)
E31	IERAPI
E35	IERRBM, IERRBO, IER8BO
E37	IERRPG
E38	IERAGK, IERAPF, IERAPK
E39	IERAGK, IERAPF, IERAPK
E61	IERROQ, IERROV

Appendix B: Register Usage

The general registers used by the sort/merge program for linkage and communication of parameters follow operating system conventions.

General register 1 is used to pass the address of a parameter list to the called routine.

General register 13 contains the address of an area set aside by the sort/merge program, in which a user routine may save the contents of registers.

General register 14 contains the address of the sort/merge program return point.

General register 15 contains the address of the user routine. It is also used by the user routine as a return code register to communicate information to the sort/merge program.

Appendix C: Messages Produced by the Sort/Merge Program

This appendix lists the messages produced by the various modules of the sort/merge program.

Message	Module Causing Message Execution
IER001A COL 1 OR 1-15 NOT BLANK	RCC
IER002A EXCESS CARDS	RCC
IER003A NO CONTIN CARD	RCC
IER004A INVALID OP DELIMITER	RCC
IER005A STMT DEFINER ERR	RCC
IER006A OP DEFINER ERR	RCC
IER007A SYNTAX ERR-xxx	RCD
IER008A FLD OR VALUE GT 8 CHAR-xxx	RCD
IER009I EXCESS INFO ON CARD-xxx	RCD
IER010A NO S/M CARD	RCE
IER011A TOO MANY S/M KEYWORDS	RCE
IER012A NO FLD DEFINER	RCE
IER013A INVALID S/M KEYWORD	RCE
IER014A DUPLICATE S/M KEYWORD	RCE
IER015A TOO MANY PARAMETERS	RCE
IER016A INVALID VALUES IN FLD	RCE
IER017A ERR IN DISP/LENGTH VALUE	RCE
IER018A CTL FLD ERR	RCE
IER019A SIZE/SKIPREC ERR	RCE
IER020A INVALID REC KEYWORD	RCG
IER021A NO TYPE DEFINER	RCG
IER022A RCD FORMAT NOT F/V	RCG
IER023A NO LENGTH DEFINER	RCG
IER024A ERR IN LENGTH VALUE	RCG
IER025A RCD SIZE GT MAX	RCG
IER026A L1 NOT GIVEN	RCG
IER027A CF BEYOND RCD	RCG, RCI

(Part 1 of 3)

Message		Module Causing Message Execution
IER028A	TOO MANY EXITS	RCH, RCD
IER029A	IMPROPER EXIT	RCH
IER030A	MULTIPLY DEFINED EXIT	RCH
IER031A	INVALID MODS OP CHAR	RCH
IER032A	EXIT E61 REQUIRED	RCH
IER033A	CF SEQUENCE INDIC E REQUIRED	RCH
IER034A	PARAM ERR FOR MODS	RCH
IER035A	DUPLICATE MOD RTN IN PHASE	RCH
IER036I	B = xxxxxx	RCK-BGA-BGB
IER037I	G = xxxxxx	RCK-BGA-BGB
IER038I	NMAX = xxxxxx	RCJ-BGA
IER039A	INSUFFICIENT CORE	RCP, RCK, RCL, RCS, RCI, BGB
IER040A	INSUFFICIENT WORK UNITS	RCI, RCJ
IER041A	N GT NMAX	RCJ
IER042A	UNITS ASGN ERROR	RCI
IER043A	DATA SET ATTRIBUTES NOT SPECIFIED	RCI
IER044I	EXIT Exx INVALID OPTION	AGA, AGG, APK, APF, AGI, AGJ, AGK, AGN, 9GN
IER045I	END SORT PH	RPC
IER046A	SORT CAPACITY EXCEEDED	AOK, ROK, ROR, ROS, ROI, RON, 8ON, RPE
IER047A	RCD CNT OFF, IN xxxxxx, OUT xxxxxx	RPC, RPF, RPG, RPM, 8PM, RON, 8ON
IER048I	NMAX EXCEEDED	RDD, RDE, RDR, RDS
IER049I	SKIP MERGE PH	RPC
IER050I	END MERGE PH	RPF, RPM, 8PM
IER051A	UNENDING MERGE	ROR
IER052I	EOJ	RPG
IER053A	OUT OF SEQ	ROP, ROQ, ROU, ROV
IER054I	RCD IN xxxxxx, OUT xxxxxx	RPG
IER055I	INSERT xxxxxx, DELETE xxxxxx	RPG
IER056A	SORTIN/SORTOUT NOT DEFINED	RCI
IER057A	SORTIN NOT SORTWK01	RCI

(Part 2 of 3)

Message	Module Causing Message Execution
IER058A SORTOUT A WORK UNIT	RCI
IER059A RCD LNG INVALID FOR DEVICE	RCI
IER060A DSCB NOT DEFINED	RC4
IER061A I/O ERR xxx	AGD, AGE, AGM, AGP, RGB, RGC, RGD, RGE, RGL, RGM, RGO, RGP, RPA, RPB, RPD, RPE, RPO, RPN, 8GB, 9GC, 8GC, 8PA
IER062A LE ERR	RCO
IER063A OPEN ERR xxxxxx	RCM,RCZ
IER064A DELETE ERR	RCV
IER065A PROBABLE DECK STRUCTURE ERROR	RCH
IER066A APPROX REC CNT xxxxxx	AOK, ROK, ROR, ROS, ROI, RON, 8ON, RPE
IER067I INVALID EXEC OR ATTACH PARAMETER	8CM
IER068A OUT OF SEQ SORTINxx	ROP, ROQ, ROU, ROV

(Part 3 of 3)

In addition to the above messages, the sort/merge program provides the facility to print diagnostic messages, control statements, and a module map. Use this option only if a problem is encountered while trying to execute the sort/merge program. This option is designed to print addresses of areas which are critical to program execution and enables qualified IBM representatives to pinpoint possible system and/or machine problems. Do not include this option in a normal sort environment; it impairs sort performance.

To print diagnostic messages, control statements, and a module map, the following specifications must be provided in the execute card:

```
                    {SORT }
//STEP1  EXEC  PROC={     },PARM='DIAG'
                    {SORTD}
```

If the DIAG message is used critical messages will give a system completion code of 0C1. If SYSABEND or SYSUDUMP DD cards are included in the job stream a storage dump will be written on this data set.

The following diagnostic messages result from using DIAG:

Diagnostic Message	Module Causing Message Execution
IER900I GENERATED CORE END ADDR xxxx	APG, APL
IER901I INPUT BFR TBL ADDR xxxx	APG, APL
IER902I OUTPUT BFR ADDR xxxx, xxxx	APA, APB, APN, 9PA
IER903I RSA TBL ADDR xxxx	APG, APL
IER904I TREE ADR FROM xxxx to xxxx	AOA, AOB, AOC, AOD, AOE, AOF, AOG, AOH
IER905I MOVE RTN ADDR xxxx	ABF, ABS
IER906I DCB TBL ADDR xxxx	AGA, AGI, AGN, 9GN
IER907I O/P CCW ADDR xxxx	APA, APB, APN, 9PA
IER908I OUTPUT IOB ADDR xxxx	APA, APB, APN, 9PA
IER909I OPEN LIST ADDR xxxx	APA, APB, APN, 9PA
IER920I GENERATED CORE END ADDR xxxx	APH
IER921I INPUT BUF TBL ADDR xxxx	APH, APL
IER922I OUTPUT BFR ADDR xxxx, xxxx	APD, APE, APO
IER923I MOVE RTN ADDR xxxx	ABR
IER924I DCB TBL ADDR xxxx	AGG, AGJ, AGP
IER925I O/P CCW ADDR xxxx	APD, APE, APO
IER926I IOB TBL ADDR xxxx	APD, APE, APO
IER927I I/P CCW ADDR xxxx	AGB, AGC, AGL, AGO, 9GB
IER940I GENERATED CORE END ADDR xxxx	API
IER941I INPUT BFR TBL ADDR xxxx	API
IER942I OUTPUT BFR ADDR xxxx, xxxx	ABL, ABM, ABN, ABO, ABP, 9BO, 9BN
IER943I MOVE RTN ADDR xxxx	ABQ
IER944I DCB TBL ADDR xxxx	APF, APK, AGK
IER945I I/P CCW ADDR xxxx	AGD, AGE, AGM, AGP, 9GC
IER961I TECHNIQUE xxxx	BGA, RCK, BGB
IER962I NO/SIZE OF BFRS, PH x, x, xxxx	BGA, RCK, BGB
IER963I MAX. SYSGEN CORE xxxx	BGA, RCK, BGB
IER964I CALC. CORE PH1=xxxx	BGA, RCK, BGB
IER965I MERGE ORDER=xxxx	BGA, RCK, BGB
IER988I IERyyy LOC. AT xxxx[1]	RC6, RC7, RC8, RC9

[1]This message will appear frequently and is designed to provide the starting addresses of the sort/merge program modules.

Appendix D: Format Codes

This appendix describes the structure of a node and the various format codes as they appear in the second word of that node.

For fixed-length records, the node consists of five words. The first word contains the address of the next-level node to which records associated with the current-level node are compared. In other words, the first-level node points to the second-level node, which points to the third-level node, etc.

The second word contains the format code. This code is a number that is used as a displacement value to index a branch table in the ordering module. The entries in the branch table reflect the sequence (old or new) to which each record at a node belongs. This knowledge precludes needless compares and facilitates the updating of a node after the position of a new record is determined.

The last three words refer to the actual addresses of the three records in the RSA. A binary compare is made to determine which word in the node is to receive which RSA address. If the user specifies ascending sequence, the address of the record having the smallest control field is placed into the first word, the address of the record having the largest control field is placed into the third word, and the address of the record having the control field that collates in the middle is placed into the second word. For descending sequences, the address of the record having the largest control field is placed into the first word, the address of the record having the smallest control field is placed into the third word, and the address of the record having the control field that collates in the middle is placed into the second word.

For variable-length records, the node consists of only three words. The first two words contain the next-level node address and format code and are functionally similar to the fixed-length record node described above.

Because of the complexity of address structuring in the variable-length record format, only one record is referred to in the RSA. Hence, only one word is required in the node for this purpose. Functionally, however, the word is similar to that in the fixed-length record node.

FORMAT CODES FOR FIXED-LENGTH RECORDS

The format codes for fixed-length records are interpreted as follows:

Format Code	Meaning
0	No record addresses in node.
16	An event has occurred: flushing completed, winner obtained, or new string started. This is a program node, as opposed to a tree node.
32	One address in the node. Record is for new sequence.
48	One address in the node. Record is for same sequence.
64	Two addresses in the node. Both records for new sequence.
80	Two addresses in the node. One record for new sequence, and one record for same sequence.
96	Two addresses in the node. Both records for same sequence.
112	Three addresses in the node. All records for new sequence.
128	Three addresses in the node. Two records for new sequence, and one record for same sequence.
144	Three addresses in the node. Two records for same sequence, and one record for new sequence.
160	Three addresses in the node. All records for same sequence.

FORMAT CODES FOR VARIABLE-LENGTH RECORDS

For variable-length records, a slight difference occurs in the format codes because only one record address is entered in each node. The various codes and their meanings are as follows:

Format Code	Meaning
0	No record address in the node.
16	An event has occurred: flushing completed, winner obtained, or new string started. This is a program node, as opposed to a tree node.
32	Describes the status of the node. (See program listing for details.)
48	Record address in the node is for a new sequence.
64	Describes the status of the node. (See program listing for details.)
80	Record address in the node is for the same sequence.

CONDITION CODES FOR FIXED- AND VARIABLE-LENGTH RECORDS

The format code determines the point of entry into the instruction sequence. When the instruction sequence is entered, one of four condition codes exists. These condition codes dictate the final disposition of the record and are as follows:

Condition	Interpretation
Flush	Force records from the tree.
Fill	Continue filling the tree.
Same	Record is of same sequence as previous records.
New	Record begins a new sequence.

Examples: Fixed-Length Records

The examples below will aid in interpreting the listings for fixed-length records. The character to the left of the slash represents the record entering the node, and the three characters to the right of the slash represent the records already in the node. Hence, in the program listing, when the comments contain a statement "This case handles an x/xxx situation," it can be resolved as follows:

Condition	Interpretation
S/SSS	New record is of same sequence as three previous records.
N/SSS	New record begins new sequence; previous three records of same sequence.
-/SSS	Records in tree are of the same sequence, and the program is in flush mode.
N/SSN	Two sequential records, one new sequence in node. New sequence record entering.
-/NNN	Flush mode; all records of new sequence to be flushed.
-/SS-	Flush mode; two records of same seqeunce to be flushed.
N/S--	One record in the node is of the same sequence; new record begins a new sequence.

Examples: Variable-Length Records

For variable-length records, the listings may be interpreted as explained below. The two characters to the left of the slash represent the record entering the node and the two characters to the right of the slash represent the record already in the node.

Condition	Interpretation
S1/S1	New record is of same sequence as previous record.
T1/S1	Record in node is of the same sequence as others in tree. Set status of node to reflect this condition.
T2/S1	Record in node is of the same sequence. New record will change sequence. Set status of node to reflect change.
S2/S2	Record in node is of new sequence. New record is also for new sequence.
T2/S2	Record in node is for new sequence. Set node status to temporary new sequence record.
T2/T1	Set node status at temporary before next record is introduced.

Appendix E: Checkpoint/Restart Facility

To eliminate the need for completely re-executing a sorting application after an I/O error, a machine check, an intentional operator interruption, or a similar event, the sort/merge program makes use of the Operating System Checkpoint/Restart facility. The user directs the sort/merge program to use this facility by (1) including the checkpoint parameter (CKPT) on the SORT control statement and (2) providing a SORTCKPT DD statement to define the checkpoint data set. (Refer to the publication IBM System/360 Operating System: Sort/Merge.)

When directed in this manner, the sort/merge program issues checkpoint macro instructions (CHKPT) at the start of the sort phase, during the intermediate merge phase (for all techniques except crisscross), and at the start of the final merge phase. The checkpoint macro instructions cause checkpoint records to be written on the checkpoint data set. These records contain information needed to restart processing.

The sort/merge program can be restarted from the checkpoint taken at the start of the sort phase or from the last checkpoint written.

The interface between the sort/merge program and the checkpoint restart facility is module IERCHK, which issues checkpoint macro instructions. The sort/merge program modules that interface with the checkpoint restart facility through module IERCHK are:

- IERAPC -- Start of the sort phase (all techniques).
- IERAPJ -- Start of the intermediate merge phase (all techniques except crisscross and oscillating). Start of each intermediate merge phase pass (balanced direct access technique).
- IERRON -- During the intermediate merge phase (oscillating technique).
- IERROS -- During the intermediate merge phase (polyphase technique).
- IERROT -- During the intermediate merge phase (balanced disk technique).
- IERAGH -- Start of the final merge phase (all techniques).

Appendix F: Program Listing Standards and Conventions

To facilitate the identification of modules, work areas, tables, and other aspects of the sort/merge program listing, symbolic names are assigned to assembler language statements according to a definite pattern.

MODULE NAMES

The format of all module names is IERTMM, where:

- IER is the identification for sort/merge modules.
- T, in general, is either an "A" for an Assignment or an "R" for a Running type module; however, for some assignment modules associated with the crisscross technique, T is "9", and for some running modules associated with the crisscross technique, T is "8."
- MM is the unique portion of the module name.

Note: Modules EX1, EX2, and EX3 do not follow the rules of name format and are used only if user-modification routines are link edited. Module DM4 also does not follow these rules and is used only if the option to print diagnostic messages is specified. Module BGA is always included for tape B and G calculations, and module BGB is always used for crisscross direct access B and G calculations. Module CHK issues checkpoint macro instructions when checkpoint is requested.

MODULE CLASSIFICATIONS

Four classifications of modules appear in the program listings as follows:

A

The operation of the module does not depend upon a particular internal representation of the external character set.

B

The operation of the module does not depend upon a particular internal representation of the external character set except that the decimal numbers are coded. The numbers are coded so that the low-order four bits, when considered as binary integer, identify the value of the digit.

C

The operation of the module depends upon an internal representation of the external character set equivalent to the one used at assembly time.

D

The operation of the module depends upon a classification of the external character set by means of a table. This table is constructed for the EBCDIC character set. The table is arranged so that the redefinition of character constants by reassembly will result in a correct table for new definitions, if the external bit remains unchanged.

INSTRUCTION NAMES

The format of all internal type instruction names is MMNNNNNO or MMMNNNNO where:

- MM or MMM represents the last two or three characters of the module name.
- NNNNN or NNNN is a unique designation assigned by the programmer and may be from one to five characters.
- O is an "X" if the label is externally used; otherwise it can be used as another N.

CONSTANT NAMES

The names of constants start with a K. The rest of the name is meaningful with relationship to some characteristic of the constant (e.g., KONEH might be used as the name of a halfword one).

WORK AREA NAMES

The names of work areas have the same format as that of constants except that the first character of the work area is a W.

TABLE NAMES

The names of tables have the same format as that of constants except that the first character of the table name is a T.

PHASE-TO-PHASE INFORMATION AREA NAMES

All references to locations within the PPI area (IERRCA) will have the format PPI nnnnn, where nnnnn is the unique designation that has been specified in the sort/merge program.

USE OF ROUTINES IN MORE THAN ONE MODULE

Some routines are used in more than one module. To permit these routines to be inserted in several modules without making any changes, the format of all internal type instruction names in these routines is SMNNNNNO, where:

- S is an "S".
- M is a unique alphameric character designated for the routine.
- NNNNN is a unique designation assigned by the programmer and may be anywhere from one to five characters.
- O is an "X" if the label is externally used; otherwise it is used as another N.

Index

Address
- chain 40,41,42
- of buffer 42
- of DCB table 93
- of disk directory 89
- of GETMAIN tables 93
- of IOB tables 93
- of unit count table 93

Ascending sequences 19
Assignment modules, list of 67-72
ATTACH 8,10,14,21

B 43,91
Balanced direct access technique 23-24,20-22,4652-54,57,58
- block/deblock modules 55,59,65
- DCB, IOB modules 56,60,66

Balanced tape technique 28-29,20-22,46,52-54,57,58
- block/deblock modules 55,59
- DCB, IOB modules 56,60,66
- intermediate merge record movement .. 36

BALN 21,22
- in parameter list 22
- (see also Balanced tape and Balanced direct access techniques)

Base level 24
Bin 39-40
- continuation 40,41
- size 41,42,91

Binary
- compare 36-37
- control fields 44

Block 47
Blocking 42,43
Block/deblock modules 38
Buffer
- address 42
- deblocking of 28

Buffer-to-buffer move 42
Build module 66

Chain address 40,41,42
Channel 20
Character control fields 44
Checkpoint
- data set 103
- modules 66
- records 103

Checkpoint/restart 103
CKPT 103
Codes
- completion 102
- format 19,101-102

Collation 19
Completion codes 102
Constant names 104
Continuation bins 40,41

Control
- blocks 43
- fields 44,87-88
- statements 8

Copy pass 29
CRCX 21,22
- in parameter list 8
- (see also Crisscross direct access technique)

Crisscross direct access technique 24-27,7,10,17,20-22,46
- block/deblock modules 56,59,65
- DCB, IOB modules 56,60,66
- read/write modules 60
- sort/merge module organization ... 61,62
- sort modules 54

Data
- management 7,36
  - routines 47
- set size 7,20

DCB table address 93
Ddnames
- modification of 8

Deblocking 28,38,42,47
Decision tables 54-55
Definition phase 7,49,50
- functions of 43
- module list 67-76

Density 20
Descending sequences 19
DIAG 79
- in parameter list 8
- Message 99

Diagnostic
- messages 8,9,99-100
- output of linkage editor 9

Direct access technique
- (see Balanced direct access technique and Crisscross direct access technique)

Directory
- intermediate storage area 23
- module 67-72

Disk directory address 89
Dummy sequences 30

End-of-file on input 17
Error messages 97-99
Equals module 44
EXCP 7,37
EXEC statement 10,13
Extract
- data 39,40
- module 44
- routine 87

E15 routine
- address in parameter list 8

E35 routine
- address in parameter list 8

E61 routine 44

F field 87
File size 91
Final merge phase 7,10
   charts 63,64
   decision tables 65-66
   determining input units for 32
   functions of 46-47
   module list 67-76
   name on SYSLMOD 9
Fixed-length records
   block/deblock modules 55,56-59,65
   completion codes 102
   format codes 101
   format in RSA 39
   merging techniques 36
   movement 39-40
   move modules 56
   nodes 101
   sorting technique 19-20
   sort modules 54
   tree structure 19-20
Flagged records 19
Flowcharts 49-66
Format codes 19,101-102

G 47,91
GET 37,40
GETMAIN tables, addresses 93

Identify matrix 30
IERAGH 103
IERAPC 103
IERAPJ 103
IERCHK 103
IERRCA 31
IERRCB 43
IERRCM 43
IERRCO00 43
IERRCV 43,44
IERRCZ 43
IERRC6 44
IERRC7 45
IERRC8 46
IERRC9 44
IERROJ 32,33
IERRON 103
IERROS 32,103
IERROT 103
Initiating sort/merge 10
Inline move 39,55
Input
   blocking 87
   buffer 17,42
      size 92
   data sets 46
Input/output
   intermediate storage 7
   modules 37
Instruction names 104
Intermediate merge phase 7,10
   decision tables 59-60
   functions of 45-46
   module list 67-76
   name on SYSLMOD 9
   sequence control 32
Intermediate storage
   formulas 22-23
   I/O operations 7
   units 21-22
IOB tables, address 93

K 39

LINK 8,10,14,21
   example 9
Linkage editor 9,16,43
Link edit 7,10
Link library 7,9,10
Listing standards 104-105
Load point, oscillating technique 33
Load routine, sort phase 44
Locate mode 37
Logical
   tape 32
   unit 32
Loser record 19

Main storage 20,22
   layout 79-82
   requirement for various techniques .. 22
   value in parameter list 8
Maximum
   input, tape 20-21,22
   merge order 23
Merge 10,15
   network 45-46,47
      prime area 87
   order 23,33,46,92
      crisscross 24
Merging technique 36,37
Messages 97-99,100
Mode 36,37
Modification routines 9
MODS statement 9,43
Module
   block/deblock 55,56,59,65
   build 66
   classifications of 104
   DCB, IOB 56,60,66
   directory 67-76
   equals 44
   extract 44
   interface list 93-94
   list 67-76
   map 79,99
   names, format of 104
   object 43
   phase control 44
   read/write 60
   sort 54
Move
   in-line 39-55
   list 39
   mode 36,37
   multiple 39,55
   variable 55,56,65

N 20
Names
   constant 104
   instruction 104

of phases in SYSLMOD 9
PPI, format of 105
Nmax 91
Nodes 17,19,101-102

Object modules, selection of 43
Optimization phase 7,9,43-44,49,51
OSCL 21,22
in parameter list 8
(see also Oscillating tape techniques)
Oscillating tape
techniques 33-35,8,10,17,20-22,46
block/deblock modules 56,59
DCB, IOB modules 56,60,66
example of 34-35
module list 67-76
read module 60
sort modules 54
sort/merge module organization ... 61,62
table 89
Output
blocking 91
buffer 17,19,36,42
size 92
of linkage editor 9
operations 46
Overlap 37
Overlay structure 73-78

Packing buffer 87
Parameter list 8,21
PARM field options 8,21
Performance, impairment 9
Phase control module 44
Phase-to-phase information area 83-94,23,43
names, format of 105
Physical
tape 32
unit 32
POLY 21,22
in parameter list 8
Polyphase tape
techniques .. 29-33,8,20-22,46,52-54,57,58
block/deblock modules 55,59
DCB, IOB modules 56,60,66
PPI 83-94,23,31
names, format of 105
PPIBDSVA 42
PPISW1 89-90,44,46
PPITPTBL 32
Program listing standards 104,105
PUT routine 36,37,40

QSAM routine 7,37

Record
area in bin 40
count during block/deblock 38
counter 91
movement techniques 39
storage area 17
RECORD statement 8
Records
fixed-length
(see fixed-length records)
flagged 19
movement of 37-42
number of 43
spanned
(see spanned records)
variable-length
(see variable-length records)
variable-length spanned 17
VRE 17
Read
backwards 28,38
final merge phase modules 65
modules 37,60
while-write, oscillating technique .. 33
Read/write modules 37
Register
save area 87
usage 96
Replacement-selection 17-18
illustration of 18
Restart 103
Rmax 91
RSA 17,19,38
Running modules, list of 67-76

Sequence
ascending 19
counters 32,91
descending 19
dummy 30
level numbers 32-33
Sequence distribution 20-22,7,9,46
modules used for 59
Size
of buffer 92
of data set 7,20,91
of file 91
SIZE operand 20
Skip record count 87
Sort
blocking 20,91
library 10
system interface 43
work area 87
Sort phase 7,9,49,52,53
decision tables 54-56
functions of 44-45
module list 67-76
name on SYSLMOD 9
sequence control 32
SORT statement 8
SORTIN 8
Sorting technique 17
SORTIN01-16 10
SORTLIB 9,10,11
SORTMODS 9
SORTOUT DD statement 10,38
SORTWK data sets 9,10
Spanned records 38-39,40,42,44
merging technique 36
SYSIN 9
SYSLIN 9
SYSLMOD 9,10
SYSPRINT 9
System generation 10,43
SYSUT1 9
SYS1.SM023 10

Tables
   for polyphase 31-32
   names 104
Tape 7,20
   logical 32
   physical 32
   switching, oscillating technique 33
   table 88
Tape techniques 28-53
   record arrangement in buffer 38
TBLPH1RN 44
TBLPH1AS 44
TBLPH2AS 45
TBLPH2RN 45
TBLPH3AS 46
TBLPH3RN 46
Tree address 39,40
Tree structure 17,19

Unit count table address 93
User exits 95
   block/deblock modules for .. 55,56,59,65
User modifications 8
   effect on record movement 39

Variable-length records
   block/deblock modules for .. 55,56,59,65
   completion codes 102
   format codes 101-102
   merging technique 36
   movement of 40-42
   move modules 56
   nodes 101
   sorting techniques 19
   sort modules for 54
   tree structure 19
Variable-length spanned records 17
   (see also VRE records, spanned records)
Variable move 55,56,65
VRE records 17,37,40,42,44
   block/deblock modules for 55,56,65
   merging technique 36

Winner record 19
Work area 7,20,21,22
   address 42
   for VRE records 17,36,40
   names 104
Work tape, oscillating technique 33
Work unit 21,22
Write modules 37,60
WSWITCH 44,45,46

XCTL 8,10,14,21

8-way merge, modules for 59,65
16-way merge, modules for 59,65
2301 7,20
2311 7,20
2314................................. 7,20
3400 7

IBM Technical Newsletter

This Newsletter No. GN33-8164
Date January 31, 1973

Base Publication No. GY28-6597-4
File No. S360/S370-33

Previous Newsletters None

OS SORT/MERGE LOGIC

This Technical Newsletter, a part of release 21 of OS, provides replacement pages for the subject manual. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

Front Cover,2
9,10
73,74

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Correction of errors.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden

Printed in U.S.A.

Reader's Comment Form

OS Sort/Merge Program Logic GY28-6597-4

Your comments about this publication will help us produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name ______________________

Job Title ______________________

Address ______________________

______________ Zip __________

No postage necessary if mailed in the U.S.A.

GY28-6597-4

YOUR COMMENTS, PLEASE . . .

Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

CUT ALONG THIS LINE

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention. Department 813 L

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Reader's Comment Form

OS Sort/Merge Program Logic GY28-6597-4

Your comments about this publication will help us produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name ____________________

Job Title ____________________

Address ____________________

____________________ Zip ____________

No postage necessary if mailed in the U.S.A.

GY28-6597-4

YOUR COMMENTS, PLEASE . . .

This SRL manual is part of a library that serves as a reference source for system analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

CUT ALONG THIS LINE

Fold Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Department 813 L

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]